UNIVERSITY OF SASKATCHEWAN

Anomaly of Existing Intellectual Property Protection for Software

A Thesis Submitted to the College of Graduate and Postdoctoral Studies in Partial Fulfillment of the Requirements for the Degree of Master of Laws (L.L.M) in the College of Law, University of Saskatchewan
Saskatoon

By

Molla Mekonen Abey

PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor who supervised my thesis work or, in their absence, by the Deans of the Colleges in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other uses of materials in this thesis in whole or part should be addressed to:

Dean
College of Law
University of Saskatchewan
15 Campus Drive
Saskatoon, Saskatchewan S7N 5A6Canada

OR

Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
105 Administration Place
Saskatoon, Saskatchewan S7N 5A2 Canada

ABSTRACT

The digital sphere, "cyberspace," is growing by leaps and bounds. Computers and programs are making a profound impact on every aspect of human life: education, work, warfare, entertainment and social life, health, law enforcement, *etc*.. So, the fact that people now need access to digital technologies to sustain modern social, economic and political life is not in dispute. Most digital devices such as computers are useless without programs. Simply stated, access to digital technologies depends highly on software. More precisely, it is practically impossible these days to find a life without the involvement of software and software-based devices. Software used to be, in the 1970s and early 1980s, applied to huge mainframe computers that took up the space of, maybe, an entire room. These days, we have software applied everywhere, in many aspects of our lives. Before the 1960s, vendors distributed and sold software bundled with computer hardware. During that time there was no clearly recognized protection for computer programs. As time went on, vendors began to unbundle software from hardware and started to provide programs to the public separately packaged.

With a view to responding to the needs of industry, on one hand, and to advancing innovation, and encouraging the dissemination of useful arts for the general public on the other, different jurisdictions began to afford separate legal protections to computer software. Many jurisdictions opted for copyright protection as the best option. We also see the widespread protection of software products by patent law. In spite of the absence of legislation which directly allows for the patentability of computer software, we witness frequent disputes and litigation as regards the scope and extent of software protection. In addition to intellectual property protections, computing companies are using technological means to exclude others from using their digital works. This approach is called self-regulation. They do so by using technology: encryption, coding, etc. It is also illegal to reverse engineer and decompile computer programs. A trade secret can be used to protect computer software, especially the inner working of software. Software developers also use the law of industrial design as another form of protection for the '*look and feel'* aspect of their software. On the other extreme, we see some movements which advocate for free and open-source software.

This thesis argues the existing system has flaws and need a fix. The main problem with existing software protection is that it overlooks its special nature. There is no dispute as to why software

is protected. Writing those millions of lines of code requires an investment of time, intellect, and money. Hence, protection is required. The issue is as to the choice of the form of protection. So, this thesis argues the blanket copyright and patent protections of software raise a fairness issue, particularly from the perspective of the consumer's interest. It also argues the existing laws governing computer software lack clarity and certainty. Overall, the thesis discusses the existing legal framework for computer programs. It concludes that the system needs reform as it mainly considers the interest of software industry. In other words, consumers and new entrants' interests have not been given much regard. More importantly, the thesis reflects on the general purpose of intellectual property rights and their applicability to computer programs. The most important reason for the reform is the unique nature of software. By doing so, the thesis suggests for the adoption of a special law for computer programs.

ACKNOWLEDGMENT

Where to begin? Although there are many to whom I owe thanks, certainly none are more deserving than my supervisor, Professor Martin Phillipson, committee members, Professor Barbara Von Tigerstrom and Robin Hansen, and College of Law Graduate Director, Professor Heather Heavin. Their contribution in editing the thesis and forwarding expertise comments were invaluable. I acknowledge with thanks, as well, to my supervisor, committee members and Professor Heather Heavin in approving funding requests towards the completion of the thesis.

Special thanks are due to Professor Barbara Von Tigerstrom, Dr. Thomas Roberts, and Lorrie Sorowski. Professor Barbara, I am grateful for your assistance. You were the one who helped me in the admission, grant permission, and deferral processes.

I would like to thank my family and friends (Violet), and special thanks to my girlfriend, Azeb Getnet, for her never failing enthusiasm and encouragement.

The unpayable debts are to my God, for his unconditional love and for giving me helpful persons in my life.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Number Page Number

INTRODUCTION

The digital sphere, "cyberspace," is growing by leaps and bounds. Computers and programs are making a profound impact on every aspect of human life:[1] education, work, warfare, entertainment and social life, health, law enforcement, *etc*. For instance, software plays an enormous role in the health sector by assisting in monitoring patients, refilling prescriptions and billing and keeping medical records. In finance, transactions involving calculations such as interest and account balances are operated by software. Air traffic control, flight schedules, booking and related tasks in the airline industry; and calculations of all sorts of incomes, benefits, expenses and interests in insurance and tax administration institutions have been undertaken with the use of software. This is just at the macro/highest level. At the individual level, the more we use digital devices, the more we need to use software to access services and products. So, the fact that people now need access to digital technologies to sustain modern social, economic and political life is not in dispute. Most digital devices such as computers are useless without programs. Simply stated, access to digital technologies depends highly on software. More precisely, it is practically impossible these days to find a life without the involvement of software and software-based devices.[2]

A computer program is a series of logical instructions to be used in a computer so that the latter produces a specific result, in the form of information. It is a technical, technological and legal concept. By "computer program", it appears we mean "programs for a computer." However, we mean more than that: software for other electronic devices, too. Software programs are useful to almost all electronic devices. The computer hardware is nothing without its software, in the form of system and application software. Other devices such as smart digital technologies, too, are helpful only with the use and application of software algorithms.

Software used to be, in the 1970s and early 1980s, applied to huge mainframe computers that took up the space of, maybe, an entire room. These days, we have software applied everywhere, in many aspects of our lives. It is not just in laptops but also on our mobile devices and is

[1] *WIPO Intellectual Property Handbook*, "Technological and Legal Developments in Intellectual Property", (2nd ed., WIPO PUBLICATION No. 489 (E): 2004) at 435.
[2] In this paper, "software" and "computer program(s)" will be used interchangeably.

increasingly integrated into all sorts of objects. We hear about the coming "internet of things,"[3] a phrase summing up the radically increasing connectivity of all sorts of items around us that, expectedly, will be communicating with each other. They will be doing so on the basis of software-based algorithms.[4] Our computers, smartphones, *etc*. are dependent for their functions on these logical instructions.

Before the 1960s, vendors distributed and sold software bundled with computer hardware. Professor Pamela Samuelson quoted the work of Justice Stephen Breyer and has stated the following: "Systems software was, 'and should continue to be, created by hardware manufacturers and sold along with their hardware at a single price".[5] During that time there was no clearly recognized protection for computer programs. As time went on, vendors began to unbundle[6] software from hardware and started to provide programs to the public separately packaged.

With a view to responding to the needs of industry, on one hand, and to advancing innovation, and encouraging the dissemination of useful arts for the general public on the other, different jurisdictions began to afford separate legal protections to computer software. Many jurisdictions opted for copyright protection as the best option. Recent international copyright treaties such as the World Intellectual Property Organization Copy Rights Treaty (WCT)[7] and the World Trade Organization Trade related aspects of Intellectual Property Right (TRIPS)[8] have a clause on the

---

[3] It is a recent agenda especially in Europe where the radical development and deployment of Internet of things technology is sought. This is with the intent to converge technologies smart environments and integrated ecosystems. See also European Commission, "Digital Economy and Society: The Internet of Things", http://ec.europa.eu/digital-agenda/en/internet-things>.

[4] Software algorithms are just rules, principles or logic by which the SW is built up on. The term algorithm and its application in software protection will be raised while discussing software patents (e.g. see, below *Benson*).

[5] Pamela Samuelson, ''The Uneasy Case for Software Copyrights Revisited", (2011) 79 Geo. Wash. L. Rev. 1746 at 1751.

[6] Yoshiyuki Miyashita, ''International protection of Computer software'', online: (1991), 11 Computer L.J. 41 at 47 <http://repository.jmls.edu/cgi/viewcontent.cgi?article=1390&context=jitpl>; Graeme Phillipson, "A Short History of Computer", (2004) , [Phillipson], at 10, [in 1968, IBM made a decision to unbundle its software for the first time and started to charge separate fee]; Peter S. Menell, "Envisioning Copyright Law's Digital Future", Online: (2002-2003) 46 New York Law Review at 73 < https://papers.ssrn.com/sol3/papers.cfm?abstract_id=328561>;Friedman, M. Mark, "Copyrighting Machine Language Computer Software-The Case Against", online: (1989) 9 Computer L.J. 1 at 4< http://repository.jmls.edu/cgi/viewcontent.cgi?article=1430&context=jitpl >.

[7] December 23, 1996, CRNR/DC/94 [hereinafter "WCT"].

[8] *Agreement on Trade-Related Aspects of Intellectual Property Rights*, Annex 1C to the *Final Act and Agreement Establishing the World Trade Organization*, December 15, 1993, 33 I.L.M. 76 (WTO). *General Agreement on Tariffs and Trade*, Uruguay Round (including GATT 1994), Marrakesh, April, 1994 [hereinafter TRIPS].

copyrightability of computer programs.[9] Obviously, it is reasonable to raise questions as to why it is not included in early copyright instruments such as the Berne Convention for the Protection of Literary and Artistic Works.[10] There were early concerns as to the inclusion of computer software in international copyright instruments. This was, partly, justified by the non-inclusion of computer software in *Berne Convention*. [11] At the regional level, too, certain jurisdictions have adopted separate copyright instruments for the protections of computer software.[12] Nation states such as the U.S.[13], Canada[14], Ethiopia[15], *etc.* also have recognized the copyrightability of computer programs. A closer look at the history of the tendency to regard software as a copyrightable subject matter tells us that the choice was not the result of research and in-depth study.[16]

We also see widespread protection of software products by patent law. In spite of the absence of legislation which directly allows for the patentability of computer software, we witness frequent disputes and litigation as regards the scope and extent of software protection. Dozens of software patents have been granted to many high-tech companies, especially in the U.S[17] and the EU.[18] The Canadian Patent Office, too, has started granting patents to software and business method inventions.[19] Even though later rejected by the European parliament, there was a proposal to adopt a law for patenting software in Europe.[20] The U.K patent office has also granted patents to

---

[9] *Ibid*, article 10

[10] *Berne Convention for the Protection of Literary and Artistic Works*, September 9, 1986, *Can T.S.* 1948 No. 22. 828 U.N.T.S. 221, revised most recently by *Paris Act relating to the Berne Convention*, July 24, 1971, 1161 U.N.T.S. 3.

[11] See, Beth Gaze, *Copyright Protection of Software* (Sydney, Australia: The Federation Press, 1989), at 189.

[12] Council Directive on the Legal Protection of Computer Programs, No. 91/250, O.J. L 122/42 (1991).

[13] US Copyright Act 1976, s. 101 [The Copyright Law of the United States of America and Related Laws Contained in Title 17 of the *United States Code*, under subject matter and scope of copyright section, defines computer program. We also have a wealth of software copyright cases battled in front of U.S courts].

[14] *R.S.C., 1985, c. C-42.*

[15] Proclamation No.410/2004 *Copyright and Neighboring Rights Protection Proclamation*, p. 2673

[16] See generally, Bessen, James E., "A Generation of Software Patents", online: (2011), Boston Univ. School of Law, Law and Economics Research Paper No 11-31 & Berkman Center Research Publication No. 2011-04, <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1868979 >.

[17] Martin Kretschmer, "Software as Text and Machine: The Legal Capture of Digital Innovation", online: (2003) JILT, < https://www2.warwick.ac.uk/fac/soc/law/elj/jilt/2003_1/kretschmer/ > [By 1999, the annual number of software patents granted in the US had risen to about 20,000].

[18] Eloise Gratton, "Should Patent protection be Considered for Computer Software- related Innovations", (2003) VII Computer L Rev & TJ at 229; *Ibid* [by 1999, about 13,000 patents covering software has been issued in Europe].

[19] *Canada (Attorney General) v. Amazon.com, Inc., [2011] FCA 127*

[20] Procedure 2002/0047/COD COM (2002) 92: Proposal for a Directive of the European Parliament and of the Council on the patentability of computer-implemented inventions.

software inventions despite its clear exclusion in the European patent convention.[21] Now we do not know what will happen after Great Britain leaves the European Union.

In addition to intellectual property protections, computing companies are using technological means to exclude others from using their digital works. This approach is called self-regulation. They do so by using technology: encryption, coding, etc. It is also illegal to reverse engineer and decompile computer programs. The famous quotation of Charles Clark- ‘‘the answer to the machine is in the machine’’[22] supports such an approach.

A trade secret can be used to protect computer software, especially the inner working of software. Such protection arises through the laws of contract and equity.[23]It is possible to enter into licensing arrangements designed to protect the trade secret in computer software.[24] Software developers also use the law of industrial design as another form of protection for the ‘*look and feel’* aspect of their software.[25]

On the other extreme, we see some movements which advocate for free and open-source software. It is based on a unique model of innovation. Basically, there are two models of innovations: the private investment model[26] and the collective action model.[27] The free software approach is different from these two models – and it is called a private-collective model. In the case of software, programmers contribute and share their knowledge, develop software and finally, leave it to the public. Free software are kinds of programs neither restricted by intellectual property rights such as copyright and patents nor by license agreements or digital right management systems. Free software can have two formats: free or open-source software. They are sometimes called FLOSS (Free/Libre/Open Source Software). When we say software

---

[21] Ronald Robertson, *Legal protection of Computer Software*, (London, UK: Longman law,1990) at 128.
[22] Charles Clark, “The Answer to the Machine is in the Machine”, in P. Bernt Hugenholtz, ed,,*The Future of Copyright in a Digital Environment*, (The Hague: Kluwer Law International, 1996), at 139.
[23] D. Jeffrey Brown & Marisia Campbell, “Copyright” in Stuart C. McCormack, ed., *Intellectual Property Law of Canada 2nd ed (New York, U.S.A: Juris Publishing, Inc., 2010)* at 46.
[24] *Ibid*, at 47.
[25] See, for instance, Dominique Nolet, “The Protection of Icons and Interfaces by Industrial Design” *ROBIC [*the visual aspect of Google’s home page is registered under the U.S. Industrial Design No. D599, 372] online: < http://newsletter.robic.ca/nouvelle.aspx?lg=EN&id=241>
[26] This is a model of innovation which allows inventors to appropriate the returns of their investment in time, money and effort. Traditionally, intellectual property system is designed to pay off such kind of inventors
[27] Collective action model is one innovation theory model that advocates for the production of public goods by giving incentives (e.g. monetary incentives). Usually, specified central agents grant those incentives such as research institutions.

is free, we mean that users can use it as they wish, modify it or fix some of its bugs, redistribute it, and access its source code.

The problem with existing software protection is that it overlooks its special nature. Software is unique. It involves the writing of millions of lines of codes in the form of source code. One can regard this part of software as a literary work and suggest copyright protection. It is true that human beings write and read books; they too can write and read source code part of the software. However, this is not the whole story. We have the compiled[28] object codes, machine-readable strings of binary numbers. It is disputable to consider those sequences of abstract algorithms as literary works. Originally, copyright protection has only been available to source code part of computer programs. Furthermore, protection is extended to the documentation and description of program codes. This is one issue with copyrighting software algorithms. Furthermore, patenting computer software raises concerns– most of which are the subject of court litigation. For instance, it is not clear whether abstract ideas, mathematical formula and theorems are patentable subject matters or fall under exclusionary clauses.

There is no dispute as to why software is protected. Writing those millions of lines of code requires an investment of time, intellect and money. Hence, protection is required. The issue is as to the choice of the form of protection. As has been said above, software is a very complicated notion. It includes source and object codes with accompanying descriptions. It could take the form of system and application software. So, the blanket copyright and patent protections of software raise a fairness issue, particularly from the perspective of the consumer's interest.

There are many threads of scholarly discourse as to whether these are the appropriate ways of protecting computer software. We have also seen disagreements between courts in connection with the protection of computer software. The existing system seems to favor only the software industry. Few scholars tend to suggest the multiple protection of computer software.[29] By doing so, they disregard the general societal and new entrants' interest as over protection denies access

[28] Source code part of software is compiled to object code using a compiler so that the computer can understand what the human programmer has written.

29 Pamela S., Randall D., Mitchel D., J.D. Reichman, "A Manifesto Concerning the Legal Protection of Computer Programs", (1994) 94 Colum L Rev 2308-2431[they suggest sui generis approach could be used with copyright, patent and trade secrets]; see also Robert A. Gorman, "Comments on A Manifesto Concerning the Legal Protection of Computer Programs," (1994-1996) 5 Alb. L.J. Sci. & Tech. 277 [hereinafter Robert].

rights of users. Others suggest the modified version of copyright to computer software.[30] Some authors have gone further and argued for *sui generis* protection as the best and better way of protecting software.[31] Such a mode of protecting software was not a novel recommendation, as the WIPO made a similar recommendation in the 1970s.[32] Needless to say, computer software requires strong protection as it is quite vulnerable to piracy.[33] However, stricter protection does not mean overprotection.

The existing laws governing computer software lack clarity and certainty. We may say copyright legislation, internationally and nationally, is regarded as the settled regulatory mechanism. However, these laws are devoid of clarity and predictability in terms of their breadth and scope. Strong criticism is and has been provided by experts[34]; courts have not yet settled the precise scope of copyright in regulating computer software.

The application of legal rules of other intellectual works to computer software without context is problematic. It is argued that the multiple protection of computer programs only serves the software industry's interest. The law [intellectual property laws] has its own justification. The utilitarian justification seems the predominant one, at least in the United States.[35] Protecting software using all the available forms of traditional intellectual property rights (IPRs) denies access to software related services than achieving the unilateral justification of (IPRs). We should not manipulate their original purpose. In the case of computer software, much of the stock of IPRs is owned by gigantic hardware companies,[36] in which case the economic incentive

---

[30] John Swinson, "Copyright or Patent or Both: An Algorithmic Approach to Computer Software Protection," (1991) 5 Harv JL & Tech 146 .The special copyright regulation of software in Europe confirms this suggestion

[31] See, *supra* note 11 at 187. She discussed the problem of adapting copyright laws, and recommended a sui generis regime as a suitable method of protecting computer programs- particularly operating system). For general understanding of this proposal, see John C. Phillips, ''*Sui generis* Intellectual Property Protection for Computer Software'', (1992) 60 Geo. Wash. L. Rev. 997

[32] Model Provisions on the Protection of Computer Software, 12 Indus. PROP.: Monthly REV. WIPO 259-73 (1977)

[33] *Supra* note 6 at 41

[34] For instance, see Laurence Diver, "Would the current ambiguities within the legal protection of software be solved by the creation of a sui generis property right for computer programs", online: (2008) 3 J Intell Prop L & Practice 2 at 126 < http://jiplp.oxfordjournals.org/content/3/2/125.abstract> .

[35] Article I, paragraph 8, cl. 8 of empowers Congress ''*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'' [*this shows intellectual property laws are designed to spur innovation and disclosure of novel ideas and works by granting limited period of exclusive right to originators of those ideas and works].

[36] John A. Gibby, "Software Patent Developments: A Programmer's Perspective", (1997) 23 Rutgers Computer & Tech LJ 293 [Most information technology firms such as IBM, Samsung, Canon, Panasonic, Toshiba and Microsoft are being awarded patents by the U.S Patent and Trade Mark Office (USPTO]

justification for software development is the weakest argument.

This thesis contains four chapters. The first chapter covers four major parts. Section I discusses definitional issues. It specifically appreciates the technological and legal meaning of computer software/programs. Section II, on the other hand, highlights the historical backdrop of computer software. Accordingly, this part outlines a very brief evolution of software. Section III appreciates the major classifications of computer software. The final part of chapter one tries to justify the legal protection of computer software.

Chapter two covers the existing intellectual property protection for computer software. It particularly discusses three forms of intellectual property rights: patent, copyright and trade secret. The chapter investigates the available laws and judicial developments in three jurisdictions and two international instruments. As a result, it examines the approaches in the U.S., Canada, and the EU. It examines legislative developments in all jurisdictions from the establishment of commissions to the adoption of laws (especially copyright). More importantly, judicial case developments regarding computer software are appreciated in this chapter.

Chapter three spells out the issue of balance of interests and some flaws of the existing form of protections. The chapter contains four parts. The first part analyzes the over-protection of computer software. It also discusses the unfair nature of the existing system, arguing the existing system disregards the interests of consumers and new entrants. Part two of this chapter specifically argues to disregard the existing system. It does that by discussing the inapplicability of the copyright, patent and trade secret to computer software. The most important part of this thesis falls under part three of this chapter. This part discusses the unique nature of computer software. For instance, the complex and omnipresent nature of software is examined in this part. Section IV concerns some balancing attempts of the existing system. On the one hand, it appreciates the doctrine of reverse engineering and public interest. It also examines the free and open source software movements and their impact on those interests disregarded by the traditional intellectual property rights.

Chapter four contains two parts. The first part provides concluding remarks. It concludes by outlining the problem of the existing system of intellectual property protection for computer software. In part two, the paper recommends the adoption of a special law for computer software.

Overall, the thesis discusses the existing legal framework for computer programs. It concludes that the system needs reform as it mainly considers the interest of software industry. In other words, consumers and new entrants' interests have not been given much regard. More importantly, the thesis reflects on the general purpose of intellectual property rights and their applicability to computer programs. The most important reason for the reform is the unique nature of software. By doing so, the thesis suggests for the adoption of special law for computer programs.

CHAPTER ONE

INTRODUCTION, DEFINITIONAL ISSUES, AND BRIEF HISTORICAL BACKGROUND

1. The Notion of Software/Computer Programs

1.1. Defining computer software

What is a computer program? Before defining "computer program", it is imperative to clarify what a computer refers to in this work. This is because the term "computer" connotes different computing devices throughout the evolution of computing technology. Computers used to include analog and digital computers in the early days. In regard to computers, this thesis only applies to digital computers which use binary digits in order to carry out their intended function. That does not mean analog computers are no longer functioning. The reason for limiting the scope to digital computers is that modern dictionaries define computer in a way relevant to this thesis. For instance, the Concise Oxford Dictionary defines computer as:

> An electronic device which is capable of receiving information (data) and performing a sequence of logical operations in accordance with a predetermined but variable set of procedural instructions (program) to produce a certain result in the form of information or signals.[37]

This being said about computers, the main issue here is programs. What is a computer program? What is the difference between computer software and program? Analysis of the existing legal protection for computer programs must begin with this definition. This question for lawyers is somewhat difficult, because if we go to the international instruments[38], we will not find an express definition of a computer program or of software and that leaves us, as lawyers, to struggle somewhat. And we struggle because it can potentially refer to a great deal, including a

[37] *The Concise Oxford Dictionary* (11th ed. , 2004), "Computer"

[38] We cannot find any reference in Bern Convention (this convention is used to be called the constitution for Copyright) about software. However, article 10(1) and 4 of TRIPS agreement and WIPO Copyright Treaty respectively define computer software. The EU software directive of 1991/2009 also defines software.

program's source code, its object code and, potentially, preparatory design materials, sketches, and drafts.[39] So we need to have a clearer understanding of what exactly is being protected under various types of regime.

Computer program is an ambiguous legal and technical concept. It is very difficult to strictly define computer software. But, in order to get its general picture let us see the literary meaning of computer program or software. The well-known dictionary for computer terms[40] defines computer program as a set of instructions for a computer to execute. A program tells a computer what to do. The term contrasts with *hardware,* which refers to the actual physical machines that make up a computer system.[41] As has been stated in the introductory section of this paper, computer programs and software are used interchangeably.

The Institute of Electrical and Electronics Engineers[42] also defines software as "computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system."[43] So, computer program or software is a set of organized instructions that guide a computer.[44]

Computer programming[45] has traditionally been an activity for trained specialists who work with pencil and paper (notionally) in the careful construction of code.[46] It is the process of translating a variety of vague and fragmentary pieces of information about a task into an efficient machine-executable program for doing that task.[47]

Computer program in isolation is nothing. It only helps the computer do a specific function(s).

---

[39] *Supra* note 12.
[40]Douglas Downing, Michael Covington, Melody Covington, and Catherine Anne Covington, *Barron's Dictionary of Computer & Internet Terms, 10th ed.,* (Barron's Educational Series: 2009) at 386 "computer program".
[41] *Ibid*, at 449
[42] This is one of the world's largest technical professional organization dedicated to advancing technology for the benefit of humanity, more information about this institute can be found at https://www.ieee.org/index.html
[43] Institute of Electrical and Electronics Engineers, IEEE Standard Glossary of Software Engineering Terminology 66 (1990) cited in Kristen Osenga, "Debugging Software's Schemas", (2014) 82:6 Geo Wash L Rev 1833 at 1836
[44] John W.L. Ogilvie, ''Defining Computer Program Parts under Learned Hand's Abstractions Test in Software Copyright Infringement Cases'', Note, (1993) 91 Mich. L. Rev 526 at 530
[45] Computer programing is the activity of writing, sequencing instructions for computers.
[46] Alan Biermann and G. Guiho, eds, *Computer Program Synthesis Methodologies: Proceedings of the NATO Advanced Study Institute,* (Bonas, France: Springer, 1982) at 335
[47] Alan W. Biermann, *Automatic Programming: A Tutorial on Formal Methodologies*, (London: Academic Press Inc., 1985) at 119

As guns do make it very easy for people to kill people, computer programs make it very much easier for people to think about the meaning of their data.[48]At the same time, the hardware by itself is of little value without the instructions that tell it what to do.[49]

As human beings use language to communicate with each other, computers use programming codes to communicate instructions. Some equate programming language with human language.[50] Of the two main defining elements of programs, ‘‘programming language’’ is one and the other is the ‘‘sequence of instructions’’. This is because, generally, programmers use different programming languages while writing millions of software instructions.

1.2. Legal/technological aspect of software

The subsequent sections address the legal definitions of computer programs.

International Level

We have many international multilateral treaties regulating intellectual property rights. Of these, few directly or indirectly address the protection of computer programs. The most relevant ones for this paper are the *Berne Convention for the Protection of Literary and Artistic Works*[51], *WIPO Copyright Treaty*[52], and the *TRIPS* Agreement[53]. Although there were attempts[54] to protect computer programs with other ways such as *sui generis* protection,[55] the most common way of protection internationally is copyright.

If we look at most legal instruments at the international level (treaties), we cannot find a direct definition of a computer program. The one exception in this regard is the EU software directive.[56] In what follows, we will examine how these instruments approach computer

[48] Eben Weitzman, Matthew B. Miles, Computer Programs for Qualitative Data Analysis: A Software Sourcebook, at 3
[49] *Supra* note 40, at 449
[50] See generally, Tutorials Point (I) Pvt. Ltd, “Computer Programing Tutorial”, Tutorials Point (2014) online: Simply Easy learning <https://www.tutorialspoint.com/computer_programming/computer_programming_pdf_version.htm
[51] *Supra* note 10.
[52] Supra note 7.
[53] Supra note 8.
[54] Supra note 26, WIPO Model Provisions
[55] *Sui generis* is a Latin phrase which means ‘of its kind.’ It means a special way of protection, as in unique, as in different from other types of IP law.
[56] Supra 12, article 1.

programs.

A recent international copyright instrument, which addresses computer programs, is the *WCT*. Article 4 of this treaty regards computer programs as copyrightable subject matter. The provision reads: *"Computer programs are protected as literary works within the meaning of Article 2 of the Berne Convention. Such protection applies to computer programs, whatever may be the mode or form of their expression."*

This section of the treaty explicitly regards a computer program as a literary work. The word 'literary' comes from the Latin "*litaritura*" (or "*litteratura*") which means written work. Perhaps, to some, the writing is confined to letters. But, it is more than letters, as the adjective "literary" must be understood as meaning all language and information-oriented productions expressed in letters, numbers or any other similar symbols, irrespective of whether they are legible for everyone or are coded (and thus available only to those who know and may use the code, or through the use of appropriate equipment).[57] This explanation is given under the section concerning the substantive provisions of the *Berne Convention*. However, we do not have an express section on computer programs in the *Berne Convention*. This may be because computer programs are recent developments and were not put on the table by signatories during the adoption (and subsequent revision) of this convention.

But, the most conceivable argument is that large computer programming industries are American. The U.S. became a party to the *Berne Convention* only in 1989.[58] Hence, there was no need to deal with computer programs as one category of intellectual property in the *Berne Convention*. This is because there were no interested groups which would bring the matter to the table.

The 1996 *WCT*, as can be seen above, referred back to Article 2 of the *Berne Convention*. This suggests the issue of computer program as a literary work should have been addressed by the *Berne Convention*. The concept of computer program was familiar in 1970s. As discussed in

[57] WIPO, guide to the copyright and related rights treaties administered by WIPO and glossary of copyright and related rights terms, (2003) , at 25

[58] Sunny Handa, *Copyright in Canada*, (Markham, Ontario: Butterworths Canada Ltd., 2002) at 158 [Major Multinational software companies like Apple Inc., Adobe Systems Incorporated, Dell, HP, IBM, Intel Corporation Microsoft Corporation, SmartZip Analytics, Superfish, Axtria, *etc.* are based in the U.S.A.]

chapter three of this thesis, it was in the late 1970s that the U.S. Congress established a commission to investigate the copyrightability of computer software. Hence, there is no conclusive justification why the *Berne Convention* has not included computer software as one copyrightable work.

The other important instrument is the *Trade-Related Aspects of Intellectual Property* agreement (TRIPS).[59] According to Article 10, "computer programs, whether in source or object code, shall be protected as literary works under the *Berne Convention* (1971). Though this clause does not directly define software, it provides that both source and object codes (which are the main elements of software) should be protected by copyright as literary works.

The U.S. used to be a pirate of intellectual property rights for centuries. This is evident from the speech of Secretary of Commerce C. William Verity while explaining the implication of the U.S.'s Policy on accession to the *Berne Convention*. It reads as follows:

"*For most of our first century of nationhood, we were takers. We stole what others created. Nobody could match us in our disdain for the rights of foreign authors such as Dickens, Thackeray, or Gilbert and Sullivan. But we soon learned that our behavior came at a cost as other nations denied our own authors the rights we had denied theirs. When nations behave that way, all of them are net losers.''*[60]

So, the U.S. has not been a party to many international intellectual property treaties until recently.[61] But when it found wealth and developed its own creative industries, then it became a leader in the adoption and promotion of even newer agreements.[62] It even used trade barriers as means to get intellectual property rights to be recognized in other jurisdictions. The 1988

[59] *Supra* note 8
[60] Orrin G. Hatch, "Better Late Than Never: Implementation of the 1886 Berne Convention", (1989), 22:2 Cornell Int'l LJ, 1 169 at 173
[61] U.S. acceded to Paris Convention on March 18, 1887, and to Berne convention on November 16, 1988 after these long years.
[62] U.S. tabled a negotiation for the adoption of international treaties which address the regulation of digital intellectual products. A case in point is WIPO copyright treaty. The inclusion of IP laws under International trade agreement is also argued to be U.S's Agenda. See generally, Pamela Samuelson, ''*The U.S. Digital Agenda at WIPO*, 37 *Va. J. Int'l L.* 369 (1996)'',available at: http://scholarship.law.berkeley.edu/facpubs/882

Omnibus Trade Act[63] is an indication of this. Simply stated, in the following section, I will discuss two international IP instruments which define computer programs.

There were model provisions that WIPO (1977) drafted to give *sui generis* protection to software back in the late 1970s. This model provision tries to define computer programs in section 1(i) as "*a set of instructions capable, when incorporated in a machine-readable medium, of causing a machine having information-processing capabilities to indicate, perform or achieve a particular result"*[64]. Computer program is defined here in terms of its function. It is a set of instructions which will make a machine (computer) work, and achieve a particular result. And those set of instructions have to be in a machine-readable form, in the form of 0's and 1's. This, in other words, means a computer cannot understand human readable source codes.

The other important point in this definitional section is the phrase *a machine having information-processing ability*. Even though the model provision is defining computer program, it did not use the word "computer". It rather uses "machine", a machine with information-processing ability. Computers have this capacity but are not unique in this regard. There are also other devices with this ability, like other special-purpose machines such as an automatic telephone exchanges and "intelligent" terminals or components thereof.[65]

According to this model provision, a distinction is made between computer program and computer software. "Computer software" is defined in a way that embraces "computer program" and "program descriptions" and "supporting material". Program description refers to a complete procedural presentation in verbal, schematic or other form, in sufficient detail to determine a set of instructions constituting a corresponding computer program. [66] This description is protected the same way computer program is protected under section 5 of the model provision. It is not a computer program *per se*. But, a computer program can be developed in a relatively

---

[63] *Omnibus Trade and Competitiveness Act*, 19 U.S.C. § 2242, 2411-2420 (West Supp. 1990) [it is an *Act* signed by President Reagan to remedy the diminishing trade-surplus of U.S.). see also Peter Clark, "A Comparison of the Antidumping Systems of Canada and the USA", (1996) at III 23.

[64] Supra note 26, WIPO Model Provisions

[65] See the comments on model provisions on the protection of computer Software, at10; sea also Patent Law - Patentable Subject Matter - Federal Circuit Applies New Factors in Deciding Patentability of a Computer Program. - *Ultramercial, LLC v. Hulu, LLC*, 657 F.3d 1323 (Fed. Cir. 2011), reh'g and reh'g en banc denied, No. 2010-1544, 2011 U.S. App. LEXIS 25055 (Fed. Cir. Nov. 18, 2011), 125 Harv. L. Rev. 2167 (2012), at p, 2172-74. In this piece the *internet* is argued to be a machine as regards computer programs is concerned.

[66] *Supra note 32 art*-1(ii)

straightforward manner.[67] The other important terminology is "supporting material". It is defined in an exclusionary manner. It can be any material, other than a computer program or a program description, created for aiding the understanding or application of a computer program, for example, problem descriptions and user instructions.[68] So, this model provision defines computer software and related terminologies unlike other instruments discussed below.

The European Union has made significant attempts to protect programs. As we will see in the coming sections, the EU has a separate law in this area.[69] The directive has some elaboration of what a computer program is, in the sense that the directive says that the term "computer program" shall embrace preparatory design material. It seems that the European Union has adopted a broad definition of computer program: "The term 'computer program' shall include programs in any form, including those which are incorporated into hardware. This term also includes preparatory design work leading to the development of a computer program provided that the nature of the preparatory work is such that a computer program can result from it at a later stage.''[70] According to paragraph 2 of article 1 of the directive, "expression in any form of a computer program" may be covered under this directive. Hence, a computer program does not necessarily cover only object and source code. It could be covering other documentation such as preparatory materials.

The EU directive seems to have adopted a broader definition than the other two international IP instruments. This can be manifested by phrases used in the preamble and article 1 such as "….*in any form……preparatory design work/material*…."

It seems there must be an intimate connection between the preparatory material and the computer program which it has prepared.

The question arises as to whether the "preparatory material" as such, independent of the

[67] See comment on model provisions- at 11
[68] Supra note 32, art 1(iii)
[69] EU has protected software with copyright since 1991(Directive 91/250/EEC) which is re- issued in slightly modified version but the change is largely cosmetic- in 2009- the directive called Software directive.
[70] *Id* recital 7; article 1(1) of the same directive also incorporated the same conception. It says ''*In accordance with the provisions of this Directive, Member States shall protect computer programs, by copyright, as literary works within the meaning of the Berne Convention for the Protection of Literary and Artistic Works. For the purposes of this Directive, the term 'computer programs' shall include their preparatory design material''*.

associated computer program, may get protection under European law. If it does, is that protection fully commensurate with the protection given to the computer program? It could be argued that the directive makes it clear that preparatory materials also enjoy copyright protection and the same sort of protection against unauthorized copying and distribution as would traditional works of copyright. Further discussion will be in order in Chapter Two as to the content of computer software protection.

But otherwise, the directive is silent on what a computer program is. Perhaps that is actually advantageous to some people. This is because, in an era of radical technological change, if we are going to define even a very basic concept like a computer program, there is a danger that we may lock the definition into a particular type of technological platform that will soon be rendered obsolete by other technology.[71] But, at the same time, it does have disadvantages, as it lacks guidance as to what exactly is being protected.

The other possible issue regarding the EU regime is the scope of protection. It is framed using open "…shall include" language. In this regard, the definition laid down in WIPO model provision seems clear. Unlike WIPO Model Provisions, the Directive uses the term "computer program" instead of "computer software". But, as we have seen above, computer software includes computer program in the model Provisions definition and structure.

In the U.S., the *Copyright Act of 1976* defines a computer program as ''*a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result"*[72] This definition is closely related to the definition of the model provision. One significant difference is that those instructions have to be used directly or indirectly by a computer, not in any machine-readable medium.

The last definitional law we will see in this paper is the Canadian approach. Section 2 of the Canadian *Copyright Act*[73] defines a computer program as ''*a set of instructions or statements,*

[71] Pamela Samuelson, "Comparing U.S. and EC Copyright Protection for Computer Programs: Are They More Different Than They Seem?" (1993)13 J.L. & Com. 279 at 282

[72] *US Copyright Act*, 17 U.S.C. (1976) s. 101.

[73] *Copyright Act* R.S.C. 1985, Chap. C-42; Unlike the case of the EU and the U.S., the *Criminal Code* of Canada also defines computer program as ''*data representing instructions or statements that,*

expressed, fixed, embodied, or stored in any manner, that is to be used directly or indirectly in a computer in order to bring about a specific result.'' The Canadian approach seems similar to the American one. The only visible distinctions are the requirements of expression, fixation, embodiment or storage of those instructions as defining elements in Canada.

We can generalize that computer program is defined in all of the above legislation in terms of its use. It is defined as a set of instructions or statements which enables the computer to produce a specific result. It is not like bells and whistles which serve a superfluous function. It, rather, enables the computer or machine-readable device to produce a certain result or solve a problem.

In the aforementioned two sections, we have seen the literal definitions of computer program. Let us conclude the definitional issues by contrasting with human language and instruction. It is like giving instruction as to the whereabouts of a specific place. If Mr. X asks Ms. Y where the College of Law is, Ms. Y will provide directions which, it is hoped, lead to the College. It can be in the form of go straight, drive a kilometer, take a right, drive around two kilometer *etc.* We apply the same logic for computer programs. Using computer programs, one gives instruction to computers or machine readable machines to perform a specific task. For instance, we can give instruction a computer to save or print our files. As humans use language to communicate directions, programmers use programming language to give instructions.[74]

2. The big bang of computer software

If we review the historical development of the computer, the Chinese created the manual operating device called the abacus in 50 BC.[75] Then, in the mid-17th century, the French mathematician Blaise Pascal invented the auditing machine by improving the abacus[76]. In 1820,

---

when executed in a computer system causes the computer system to perform a function [see Criminal Code, R.S.C.1985, c. C-46, sub-section 342. 1(2)].

[74] As we can use various kinds of languages to give directions (English, French, Spanish, Chinese), programmers also use different programming languages to give instructions to computers or machine readable devices. The most common programming languages are Java, C, C++, Python, PHP, Perl, and Ruby. See generally, Computer Programming Tutorial Simply Easy Learning by tutorialspoint.com, <http://www.tutorialspoint.com/computer_programming/computer_programming_tutorial.pdf>

75 J. B. Dixit, Sangeeta Dixit, Fundamentals of Computer Programming and Information Technology, (India: Laxmi Publications, 2005) at 11; see also Saylor Foundation, "Brief History of Computer Systems, Software, and Programing", at <http://www.saylor.org/site/wp-content/uploads/2014/07/CS101-1.1-Brief-History-of-Computer-Systems-Software-and-Programming.pdf> (hereinafter "Saylor")

76 Ibid

another French engineer greatly improved the previous adding machine and produced the multiplying machine.[77] An English mathematician and computer pioneer Charles Babbage began developing the first general purpose computing machine called the 'Difference Engine'.[78] Fifteen years later, he proposed the other general purpose computer concept. Babbage called this machine 'Analytic Engine'. Unfortunately, his idea of building these programmable engines was never successful during his lifetime because of funding.[79]

Ada Lovelace, the world's first programmer, published a paper in which she demonstrated how Babbage's analytical engine could be programmed to perform various computations.[80] Her description is now regarded as the world's first program.[81] Later in the 1970s the U.S Department of defense developed Ada Programing Language.[82]

In the 1950s, FORTRAN ('Formula Translator') and COBOL ('Common Business Oriented Language') were released, and other programming languages such as BASIC ('Beginner's All Purpose Symbolic Instruction Code') also became popular.[83] Soon, the term "systems analysis" came to be used to describe the process of collecting information about what a computer system was intended to do, and the codification of that information into a form from which a computer program could be written.[84]

The history of software directly relates with the history of hardware or computing in general. David Hayes'[85] graphical description of computer and other emerged technologies help us better grasp of the evolution.

77 Ibid
78 Graeme Phillipson, "A Short History of Computer", (2004) at 2 (hereinafter "Phillipson"); see also Saylor at 3
79 Saylor, at 3
80 Ibid
81 Phillipson, at 3
82 See, "History of the Ada Programming Language", <http://cs.fit.edu/~ryan/ada/ada-hist.html>
83 Phillipson, at 7
84 Ibid
85 David Hayes, ''Brief History of Software: From main frame to mobile'', Software IP: The 20th Annual BCLT/BTLJ Symposium – Intellectual Property Protections for Computer Programs Past, Present, and Future delivered at The 20th Annual BCLT/BTLJ Symposium of U.S, UC Berkeley School of Law, April 14th, 2016) [unpublished].

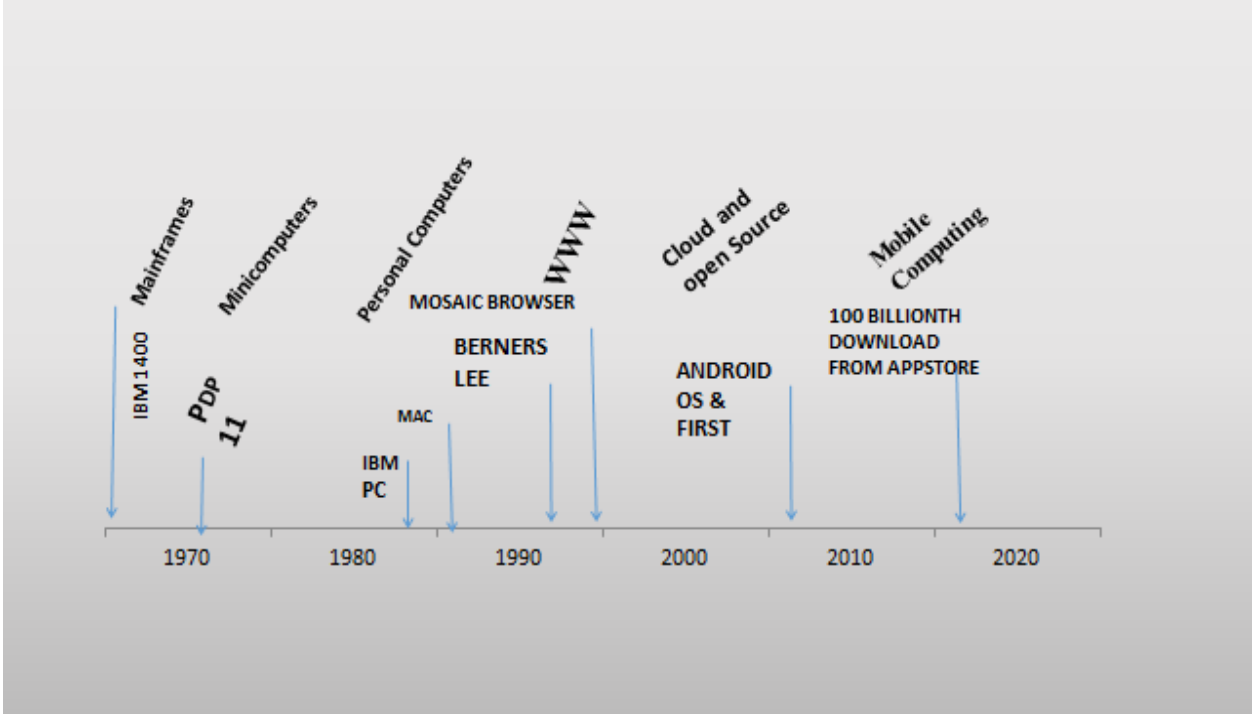


Figure 1-1. History of computing

As can be seen from the above figure, the 1960s were dominated by mainframe computers. In 1959, IBM released its first transistor-based system called IBM 1400.[86] Then, in 1970, Digital Equipment Corporation (DEC) released its Programmed Data Processor (PDP 11) ushering in the decade of minicomputers.[87] The 1980s saw the rise of personal computers.[88] IBM released IBM PC in 1981, and in 1984 Apple released the first Macintosh (Mac).[89] In 1990 Tim Berners-Lee published a formal proposal for the hyperlink world wide web.[90] In 1993 the Mosaic web browser was released.[91] The first decade of the 2000s saw the rise of cloud computing and the entry of open source into the industry.[92] In late 1990s Salesforce, a cloud computing company,

---

[86] Mike E., John K. Wayne O., and Bill O.**, "***Introduction to the New Mainframe: z/OS Basics"*, (02 January 2012), online IBM Readbooks < http://www.redbooks.ibm.com/abstracts/sg246366.html?Open>
[*This was the first mass-produced digital, all-transistorized, business computer that could be afforded by many businesses worldwide*]
[87] Gordon Bell, "Stars: Rise and Fall of Minicomputers" IEEE Xplore (17 March 2017), online: Engineering and Technology History Wiki http://ethw.org/Rise_and_Fall_of_Minicomputers >.
[88] *Supra* note 86.
[89]*Ibid.*
[90] *Ibid.*
[91] *Ibid.*
[92] *Ibid.*

launched the first commercially successful software as a service, entirely browser based.[93] In 2007 Google released the Android OS and open source license.[94] In the same year, Apple released the first iPhone. The current decade could be labeled as the decade of mobile computing.[95] In June 2015, Apple announced the 100 billionth download from its app store online.[96]

Luanne Johnson[97], in her early article nicely explained the history of computer software as follows:

''Software products are as readily available as music CDs or videotapes are to consumers today, so it is almost inconceivable that only 40 years ago the concept of software as a commercial product was considered harebrained. Yet that was the case in the 1960s. Computer users had limited choices for acquiring the software they needed to run their applications. They could obtain generalized programs from their hardware vendor at no cost because the cost of software was bundled into the computer's cost. Their second choice was to create, at great expense by using their own programmers or a contract programming firm, customized programs designed to their own specifications. Software was either free, obtained from the computer manufacturer, or customized for use by a specific customer only. Consequently, it seemed an impossibility to design software generalized enough to be sold to multiple users yet differentiated enough from the hardware manufacturers' free software that customers would willingly pay for it. The 1960s were boom years for entrepreneurial firms established to sell programming and system design skills under contract in a market where the rapidly expanding use of computers created a high demand for those skills….

…These firms increasingly found opportunities to package the software they had already written and deliver it to multiple customers, a situation that promised potentially high profits given the low cost to reproduce already developed software. The term software packages appeared in the

[93] *Ibid.*
[94] *Ibid.*
[95] *Ibid.*
[96] *Ibid.*
[97] Luanne Johnson, "*Creating the Software Industry Recollections of Software Company Founders of the 1960s"*, IEEE Annals of the History of Computing, (07 August 2002), IEEE Xplore Digital Library at 14< http://ieeexplore.ieee.org/document/988576/>

late 1960s and implied that the customer deliverables included documentation and some level of service, such as installation, as well as the program code.
Many early products were utility programs with greater functionality or efficiency than the comparable free software from the hardware vendors. Other early products were software applications like payroll or banking where external factors such as government regulations imposed uniformity on the way that customers defined their specifications.
In January 1967, International Computer Programs (ICP) in Indianapolis, Indiana, began publishing a quarterly catalog of computer programs available for sale, and the software product industry began to take shape. In June 1969, IBM announced that, effective 1 January 1970, it would charge for some of its software. Other hardware manufacturers followed suit, ending customers' expectations that generalized software would always be free and setting the stage for independent software vendors to become a significant source of software products by the mid-1970s…… about 200 companies were selling, or developing, software before IBM's unbundling took effect.''

Hence, in early days, consumers acquired software bundled into computer hardware and vendors were not charging a separate cost for software. Consumers, however, did not have an opportunity to choose specific software applications. Companies completely shifted their software manufacturing and distribution strategy. The change resulted in extra cost and choice to consumers, and commercial success to companies.

3. Taxonomy of computer programs

There are many ways of classifying computer programs. Hence, in order to capture the nature of computer programs in a simple way, it is advisable to consider these typologies. We can classify computer programs, firstly, as application or system software. Secondly, computer programs can be classified as free or proprietary. One can also categorize computer programs as program source code or object code. A key distinction is the difference between a computer program and computer software. Separate discussion of these classifications is in order.

3.1 Application and System Software

Both application and system software are instructions and statements as defined under WIPO

model provisions, and U.S. and Canadian copyright acts. Application software specifically directs the computer hardware to perform a specific or general function and helps users do specific activities using a computer's hardware. For instance, it helps users create documents using a word processor (*e.g.,* MS Word**)** or spreadsheet[98] (*e.g.,* MS Excel**)** which computes numerical tasks, or a video editor (*e.g.,* Virtual dub), or performs some other function such as AVG anti-virus, FireFox internet browser, VLC Media player and CamStudio, which screen records, and Skype, which helps to make video conferencing.

System software, on the other hand, helps a computer run properly. It controls and supports the hardware system. System software does not perform a specific function that is transparent to a user.

System software may be referred to as operating system programs (OSPs). OSPs (*e.g.,* Windows), manage the internal functions of computers, and application programs (*e.g.,* Microsoft Word and other word processing programs) perform specific data-processing tasks for users.[99]

In general, a computer may not need more than one system software program, whereas users can and often do use numerous application software programs, depending on what functions they want to accomplish.

3.2 Free and Proprietary Programs

We can also classify computer programs as free or proprietary on the basis of how they are made accessible to users; *i.e.*, whether they are provided freely or for fee.

Proprietary software is the kind of software which usually is protected by intellectual property laws.[100] Users are required to pay a fee, usually in the form of a license, to access the software. This is because software is seldom sold. In many cases, program-producing industries want to

[98] See *Lotus Dev. Corp. v. Paperback Software Int'l*, 740 F. Supp. 37 (D. Mass. 1990). *(This case involves application program software protection).*
[99] Alan Story, "Intellectual Property and Computer Software: A Battle of Competing Use and Access Visions for Countries of the South", (ICTSD and UNCTAD, 2004) at 12; see also *Apple Computer, Inc. v. Franklin Compute Corp,* 714 F.2d 1240 (3d Cir. 1983), *rev'g* 545 F. Supp. 812 (E.D. Pa. 1982), 714 F.2d 1240 (3d Cir. 1983), *rev'g* 545 F. Supp. 812 (E.D. Pa. 1982).
[100] Ibid (Alan) at 4.

transfer their software by a so-called "end user license agreement". They are not willing to sell software.[101] And they can include many restrictions in agreements of this kind.

At the other extreme, we have free software. These kinds of programs are neither restricted by intellectual property rights such as copyright and patents nor by license agreements or digital right management systems. Free software can have two formats: free or open-source software. They are sometimes called FLOSS (Free/Libre/Open Source Software). When we say software is free, we mean that users can use it as they wish, modify it or fix some of its bugs, redistribute it, and access source code.

3.3 Program Source Code and Object Code

The *TRIPS* agreement in Article 10(1) provides for protection of computer program whether in source or object code. Other instruments do not contain such a distinction. When we say that a computer program is in (expressed in) source code we mean software in a human-readable form. Programmers develop such programs in way we humans can understand (at least those of us who are computer programmers).

On the other hand, references to programs in object code mean machine-readable forms. They are expressed in binary digits, a string of 0's and 1's. Human beings, even experts on the area, cannot grasp and remember these machine codes.

A statement or mathematical expression can be made using a high-level programming language (*e.g.*, FORTRAN, BASIC, and PASCAL). And these high-level codes can be directly translated to machine code using a translator program. However, it must be observed that it is possible to translate first to **assembler** code before machine/object code. Then we need to have an assembler (assembly language) to translate from **assembler code** to **object code**. We can also automatically translate using a **compiler** (source code to object code). To make things even clearer, we can have an interpreter whereby a **translator** immediately translates so that the user

[101] This matter attracts attention if we consider digitally distributed software. Once owners of software transfer software in the form of a sale, they will no longer control the further distribution of that specific software. As the further redistribution of that software will have detrimental effect on the interest of the original sellers, the later will opt to license rather than sell. However, recently the highest court of EU in its controversial UsedSoft decision, equated software licenses with sales so that the doctrine of exhaustion applied to the transfer of software via license. See below *UsedSoft GmbH v Oracle International Corp*, Case C-128/11).

understands automatically. This is just to show how one language level can be translated to another level.[102]

Computer programs may also include some documentation. In the wording of the WIPO model provisions, this documentation includes program description and supporting materials. In EU terminology, this may mean preparatory design materials or works.

4. Justifying the protection of computer programs

Justifying IP status for computer programs is directly related to the justifications of intellectual property rights (IPRs) in general. IPRs may be justified by labor, utilitarian or personality theories.

Seen from the point view of labor theory, IPRs protect computer programmers'[103] efforts or labor. Labor-based legal theory was originally developed by the English philosopher John Locke to justify tangible property rights. However, subsequent commentators extended the application of this theory to intangible property rights such as IPRs.

Locke, in the chapter entitled 'Of Property' in his 1690 book[104] explained the basis of property as follows.

''*The earth and everything in it is given to men for the support and comfort of their existence. All the fruits it naturally produces and animals that it feeds, as produced by the spontaneous hand of nature, belong to mankind in common; nobody has a basic right—a private right that excludes the rest of mankind—over any of them as they are in their natural state. But they were given for the use of men, and before they can be useful or beneficial to any particular man there must be*

102 See generally, Hugh Brett and Lawrence Perry, *The legal Protection of Computer Software*, (Oxford, UK: ESC Publishing Ltd, 1981), at 5-11

[103] A programmer is a person who prepares instructions for computers. By and large, programmers are natural persons though there are a wider instance of cooperation, and assignment- assigning to the employers.

[104] John Locke, *Second Treatise of Government*, Book II, Ch. V , 1690, at para 26 & 27 <http://www.earlymoderntexts.com/assets/pdfs/locke1689a.pdf >

some way for a particular man to appropriate them... Though men as a whole own the earth and all inferior creatures, every individual man has a property in his own person [= 'owns himself']; this is something that nobody else has any right to. The labor of his body and the work of his hands, we may say, are strictly his. So when he takes something from the state that nature has provided and left it in, he mixes his labor with it, thus joining to it something that is his own; and in that way, he makes it his property''.[105]

As asserted in the paragraph above, Locke reasoned that individuals have, by "natural law", a property right in their bodies and, consequently, in the fruits of the labor produced by their bodies.[106] Thus, through labor, an individual converts the raw material of nature into private property, whether tangible or intangible.[107] Computer software, being one category of intellectual objects, can also be justified by this theory. In this case, our basis for protection will be that programmers own their efforts and so also the products of their efforts.

Be this as it may, some case laws seem to disregard the labor theory. For instance, in the *FEIST PUBLICATIONS*[108] case, the court said: *"The primary objective of copyright is not to reward the labor of authors, but to promote the Progress of Science and useful Arts."*[109]
A related basis is the personhood theory. Theorists argue that intellectual objects are the extensions of the creators' or inventors' personalities. By this theory, a work or invention is an embodiment of the personality of the creator.[110] Hence, a programmer by developing software is not intending to make profit or earn; he rather does it for personal development and growth.

---

[105] *Ibid.* paragraph 26 and 27
[106] Deborah Tusssey, *Complex Copyright: Mapping the Information Ecosystem*, (England: Routledge, 2012) at 42
[107] Simon Stokes, *Art and Copyright*, (Oxford, UK: Hart Publishing , 2012) at 18; Law Society of Upper Canada v CCH Canadian Ltd., [2004] 1 S.C.R. 340; rev'g (2001), 18 C.P.R. (4th) 161 (F.C.A.); allowing in part (1999) 2 C/P.R. (4th) 129 (F.C.T.D.); affirmed by Robertson v Thomson Corp. (2006), S.C.J. No. 43 (S.C.C.) [The court in determining the originality of copyrightable works, seems to incorporate labor theory]
[108] *FEIST PUBLICATIONS, INC. v. RURAL TELEPHONE SERVICE CO., 499 U.S. 340 (1991)*
[109] *Ibid*, par 19.
[110] See Tanya Alpin & Jennifer Davis, *Intellectual Property Law: Texts, Cases and Materials*, (New York; Oxford University Press, 2009) at 52 [for them, unlike the economic arguments for [IPRs], IP law regime exists, not to advance the common will, but to give force to certain ethical obligations owed to creators or [inventors]). Continental law systems seem to give a pedestal position for moral right theory in justifying copyright. See, for instance, Directive 2001/29/EC Of The European Parliament and of the Council of 22 May 2001 on the harmonization of certain aspects of copyright and related rights in the information society [This is manifested from its recital 11, which is formulated in the following manner:
''A rigorous, effective system for the protection of copyright and related rights is one of the main ways of ensuring…..of safeguarding the independence and dignity of artistic creators and performers''.]

Another justification could be economic incentive-based theory. It is otherwise called utilitarianism. This theory emphasizes on the duty of society to reward creators. The assumption of this theory is that there will be an incentive to produce goods because their selling prices will allow a producer [creator or inventor] to recoup both costs of production and the benefit of the goods to a purchaser.[111] For economic theorists, the intended beneficiary of the [intellectual object] is the community as a whole, which demands production of and access to as many creative works as possible.[112] In the case of computer programs, besides being beneficial to society they are expensive to develop. The painstaking process of formulation, coding and testing a new program requires much valuable time.[113]

Which of the above justifications have been incorporated into software laws? In the U.S., there is a constitutional clause which serves as the basis for intellectual property protection.[114] We cannot find specific justifying clauses for computer programs and other traditional intellectual objects. On the other hand, software warrants protection as its development requires the investment of considerable human, technical and financial resources, it plays an important role for community's industrial development,[115] and it can be easily exploited by others in the absence of property such as that provided through the creation of IPRs. It seems that the EU adopts the utilitarian justification.[116]

Intellectual property rights and computer software may be justified either by natural rights, labor, moral right, and personality theories on the one hand, or utilitarian theory on the other. However, all of these theories are not without their critiques. In what follows, let us see the critics posed to these theories.

Alternative ways of rewarding

The first question one can ask is why IPRs [for computer programs] at all? Is there no other mechanism of rewarding creative minds? The counter argument, of course, will be that other

[111] *Ibid*, (Tanya Alpin & Jennifer Davis) at 52.
[112] Ibid
[113] David Bender, "Trade Secret Protection of Software", (1969-1970) 38 Geo. Wash. L. Rev. 909 at910; see also Barron's Dictionary of computer Terms at 449.
[114]Article I, paragraph 8, cl. 8 of empowers Congress ''*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''*
[115] Supra note 12, recital 2 and 3
[116] *Ibid*, recital 3.

systems of rewards are not as effective as IPRs. The problem with this counter argument is we have not tried them. However, some scholars believe that in the absence of IPRs, markets [software markets] will fail.[117] For these persons, somebody may invent or create useful works. However, these intellectual objects will be under-produced unless the law intervenes to cure this 'market failure'.[118] Nonetheless, for Hettinger it is also equally important to think of alternative ways. For instance, we can use awards, acknowledgements, and public finance support systems to spur innovation and creativity, rather than IPRs.[119] Kremer also proposes government buyout of patent after conducting auction.[120] Government sponsored cash rewards as partial or full replacements of the patent system are also considerations. This is even important to address in fields where the disparity between average cost and marginal cost is typically large – biotechnology and computer software.[121]

Some Works created without expectation of IPRs

Do we have to treat all works/inventions equally? For one thing, it has been argued by many that all works are not the result of 100% individual effort. At times, the contribution may be negligible.[122] The other conceivable reason is some people may write a book or a program for their personal pleasure. Did Shakespeare write his works for incentive or IPRs? How about people who create for religious purpose or other causes? Richard Stallman's did not develop GNU software for commercial success. The same holds true to Linus Torvald's *UNIX* type *Linux* operating system. These examples show that some intellectual works could be created without consideration of IPRs.

---

[117] Supra note 29 at 2382 [they explain the market destruction concept in the context of cloning of cloning programs]

[118] *Supra* note111 at 56

[119] Edwin C. Hettinger, "Justifying Intellectual Property", online: (1989) 18 Philosophy & Public Affairs 1 at 41, 49 https://www.jstor.org/stable/pdf/2265190.pdf

[120] Michael Kremer, Patent buy outs: A mechanism for Encouraging Innovation 113Q.J.Econ.1137(1998) cited in James E Daily and F. Scott Kief, *Perspectives on Patentable Subject Matter*, (New York, U.S.A.: Cambridge University Press, 2015) at 407

[121] See especially, Steven Shavell and Tanguy V. Ypersel, "Rewards Versus Intellectual Property Rights", (2001) 44 J.L & Eco.525 (these authors argue that intellectual property rights may not always be an advantageous system and they suggest government reward system as an alternative stimulating mechanisms)

[122] For instance the required individual creation expected from author of copyrightable in U.K was very minimal. On the other hand, the extent of creativity was very high in Germany. With the view to harmonize copyright laws in Europe, the EU later adopted a copyright directive that is applicable all over Europe. See also *CCH Canadian Ltd. v. Law Society of Upper Canada*,[2004] 1 S.C.R. 339, 2004 SCC *;* Supra note 29at 2380 [software developers often consult well known program elements while writing source codes of their own]

Absence of Scientific Evidence

Even if there is an assumption that IPRs will encourage innovation, so far there is no research which shows a direct relationship between intellectual property rights and economic incentive. Simply because there is a strong belief, are we supposed to grant all creators powerful patent monopoliesand perpetual copyright protections? So, in the absence of conclusive evidence or research, it is not logical to restrain the public from freely using and commercializing their ideas. In the U.S, a 1966 presidential commission on the patent system recommended that patents should not be permitted for software, as satisfactory growth in the industry had taken place in the absence of patent protection.[123]

Hence, one cannot find a single theory fully justifying the existing system of IP law. The combination of these theories may be a better alternative. We can say the existing IPRs may be grounded by a combination of labor theory, personhood and economic incentive theories.

The EU, U.S, and Canadian intellectual property law regimes in one way or other incorporated these theories. Be that as it may, there are grievances both from the rights holders' and consumers' sides. For instance, the Pharma industry wants the further extension of patent protection.[124] There is also widespread piracy of copyrightable works for which the authors of these works look to the public and the government for assistance.[125] Authors, inventors, the publishing and recording industries restrict the free flow of information using IPRs. On the other hand, the public at large considers information as a basic necessity.

Because of these contentions, it is very difficult, if not impossible, to apply IP laws as they are to computer software. Expansion of technology exacerbates the enforcement problem even more. As the legal system is not effectively protecting their interests, the industries are devising self-enforcement mechanisms. This is true for digital intellectual objects. The copy and print control Digital Right Managements Systems (DRMS) employed by high tech corporations are an

---

[123] Supra note 17.

[124] Barrie McKenna, "Canada needs tougher drug patent protection: Report" The Globe and Mail (23 August 2012) online: The Globe and Mail < http://www.theglobeandmail.com/report-on-business/canada-needs-tougher-drug-patent-protection-report/article562405/> ; Tom Roberts, Intellectual and Industrial Property I: Introduction to Patents, Lecture Notes, (College of Law, University of Saskatchewan, 2015)

[125] See generally, Adrian Johns, *Piracy the intellectual property wars from Gutenberg to Gates*, (Chicago: The University of Chicago Press, 2009);

example. They are going in the direction of "the answer to the machine is in the machine"[126] approach. Simply stated, if we have a legal system that is based on fairly reasonable justifications, we might not encounter such a problem.

A simple premise can be made. That is, intellectual property rights are bargains between the right holder and users. These bargains have to be fair. By fair, I mean the right holder shall get what they deserve. This again will be the other premise. The third premise might be the users' right. They shall have the right to access created works in a fairly reasonable manner. Of course, it is very difficult to balance these two interests. So we have to come up with a plausible conclusion. In what follows, I will forward my solutions to this basic contention.

Firstly, we have to see each category of intellectual objects separately. A "one size fits all" approach is the heart of the cause of contention between users and holders. It is true that there exists a separate rule for patents, copyrights, trademarks, and others. Nonetheless, there are varieties of protectable subject matters in each of IPRs regimes. If we take copyright, there are expansive lists of subject matters which are copyrightable. Programmers of software and writers of songs shall be treated differently. The term of protection, breadth and scope of rights of these authors should not be the same. This is so, without derogating the very principle of labor theory or utilitarian theories. The same is true in determining the scope and duration of the exclusive right of holders of audio-visual and dramatic work. In the same fashion, patentable subject matters shall also be seen on a case by case basis. Hence, irrespective of the form of protection for computer programs, the justification should be seen in context.

Secondly**,** patents are by their nature very strong. They also preclude parallel inventions. For this reason, their term of protection is short. But it is unfair to prevent individuals, at least, from using their ideas for themselves. We can ban them from commercializing their ideas as it is not novel and somebody else is already making it available to society. But one cannot see the reason for stopping them from personally using it.

The third and most important point which is often overlooked is that the monopoly right over intellectual objects may not actually benefit the right holder. In the case of copyright, the interest

[126] Supra note 22 (as mentioned above, recent copyright instruments protect technological protection mechanisms and outlawed any attempt to circumvent those methods.

of publishing and distributing companies is not less important. The proposed reward may not ultimately benefit the actual author. A person may write a book and sell it to the public. That person then will share the net sale with those companies involved in publishing and distributing the work. My proposal for this is the state may finance these publishing and distributing industries so that the interest of the author and public at large will be reconciled. This is because publicly funded publishing organization will not have profit motive. This way consumer of copyrightable materials pays only the authors of works. The same is true for patents. Patent application is very expensive. Even after the grant of a patent, if there is any challenge, defending it is also very expensive.The consumer then bears the cost. The possible solution can be simplifying the patent grant procedure without compromising the essential purpose of a patent.

CHAPTER TWO

EXISTING INTELLECTUAL PROPERTY PROTECTION FOR COMPUTER SOFTWARE

It has been about six decades since computer software came to affect our lives. As has been explained in the preceding sections, at first we did not have a separate protection for software. We rather considered computer software as part of the general notion of a computer. So, any price we put on and protection granted to computers includes computer software. Software applications other than computer software were unthinkable 20 years ago, let alone in the 1950s and 60s.

However, discussion as to splitting software from hardware and requiring separate protection was put on the table in 1967, at least at the United Nations (UN) level.[127] The first idea was to protect software with a special law. To conduct a thorough study and come up with a feasible solution for this issue, an international committee was established. The committee prepared a model law, though it was finally rejected, and a completely new approach has been adopted.[128]

Nowadays, one can protect software in various ways. In what follows, discussion is presented as to patent, copyright, trade secret and other forms of protection of computer software. The interpretations of laws and cases by patent offices and the judiciary as to the patent protection of software are also part of the discussion in this section.

2.1 Patenting Computer Software

Computer programs were not originally considered patentable, since they were viewed as mathematical discoveries by some and abstract ideas by others. Be that as it may, today patent law is used as one way of protecting computer programs in many jurisdictions. So, in many

[127] Supra note 6 (Yoshiyuki), at 47.
[128] *Ibid.*

countries (notably in the United States), copyright was no longer the only way to protect software. Nowadays, patent law is becoming increasingly a way of protecting software in some parts of the world.[129]

As compared with the U.S. system, the EU and Canadian systems are more reluctant to grant patents for computer programs.

2.1.1 Software patents in the U.S.

Patent laws of certain countries excluded computer programs[130] in an explicit manner. However, we cannot find an explicit exclusion for patenting computer programs in the United States and some other jurisdictions.[131]

Although the U.S. *Copyright Act* explicitly regards computer programs as literary matter and, hence, copyrightable, the following figure shows how the patent system also affords protection to software and computer-related inventions.[132] According to this figure, one can understand that, although the United States Patent and Trademark Office **(**"USPTO") has no classification specifically directed towards software and computer-related inventions, it does try to quantify how many "software" patents it issues each year, stating that as many as one-half of the nearly *250,000* patents issued annually are directed towards software inventions.[133] Similar studies from the University of Edinburgh show that in a single year, the patent office granted 41,144 software patents, where the total number of patents granted in that year was 336,643.[134] Within 20 years, the number of software-related patents in the U.S. grew from 3,078 to 41,144.[135]

---

[129] Working group on Libre Software, ‘‘Free Software / Open Source: Information Society Opportunities for Europe?’’, EU commission Community Research and Development Information Center (23 February 2000), Online: EU Commission News & Events at 22 < http://cordis.europa.eu/news/rcn/14374_en.html> ; see also supra note 40at 367. In recent years, however, software patents have become common on the ground that software can be an essential part of a machine.

[130] For instance, see article 52(2) of European Patent convention. As will be discussed in the subsequent sections, what is excluded in this convention though is ‘‘computer program as such’’

[131] See, for instance David Bainbridge, “Court of Appeal Parts Company with the EPO on software patents”, (2007) 23 Computer L & Sec R at 199 (Japan’s and Australian Patent Acts have no such exclusionary provision).

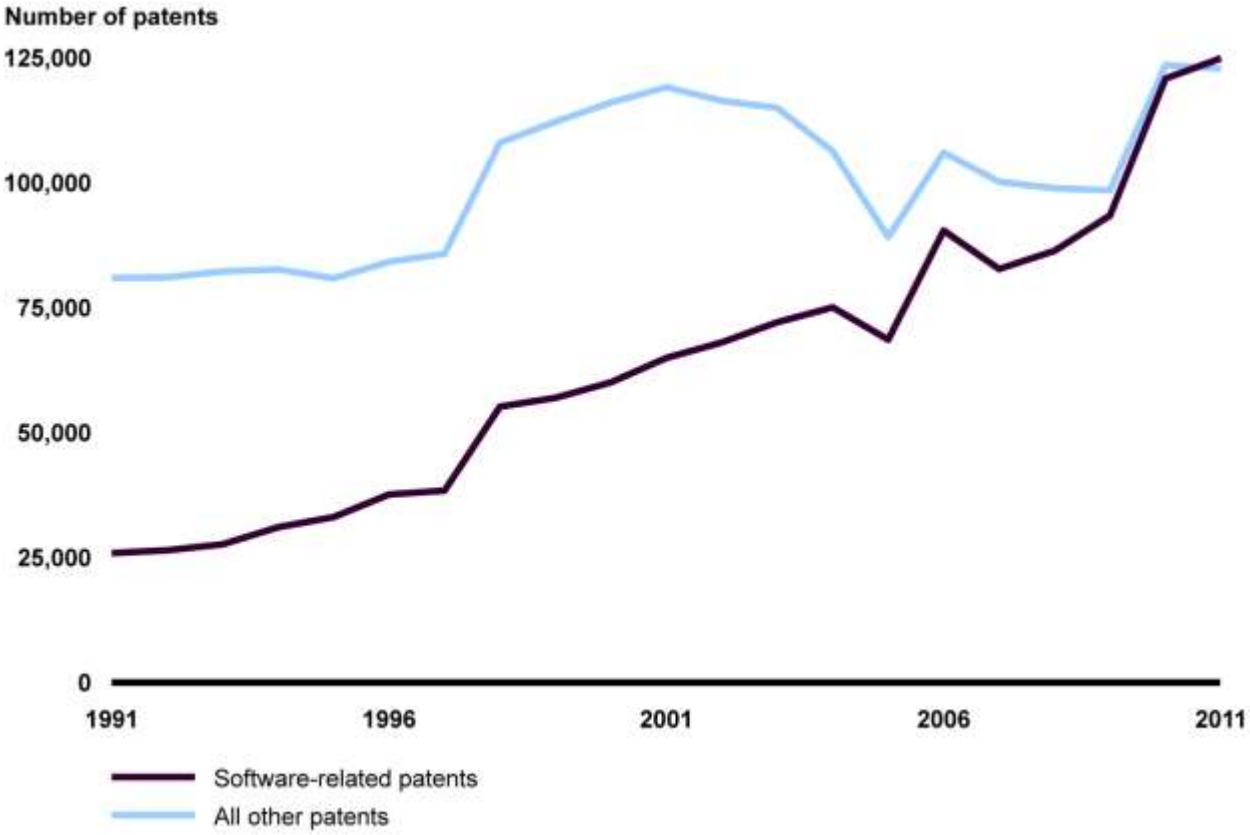
[132] U.S. Gov't Accountability Office, GAO-13-465, Intellectual Property: Assessing Factors That Affect Patent Infringement Litigation Could Help Improve Patent Quality 12 Fig.1 & N.27 (2013).

[133] Supra note 43at 1835-1836

[134] Andrés Guadamuz González, Software Patentability: Emerging Legal Issues, IP and Software (06 December 2008), online: WIPO Magazine <http://www.wipo.int/wipo_magazine/en/2008/06/article_0006.html>

[135] *Ibid*, in 1986, the USPTO has issued around 3078 software related patents; James Bessen and Robert M. Hunt**,** **“**An Empirical Look at Software Patents”, online (2007) 16:1 Journal of Economics & Management Strategy at 158 < http://onlinelibrary.wiley.com/doi/10.1111/j.1530-9134.2007.00136.x/epdf> .

The following figure explains the trend in software patents in the U.S. It is strong evidence showing the exponential growth of the USPTO's granting software patents, although we see an oscillating position between courts in the subsequent paragraphs.



Source: GAO analysis of United States Patent and Trademark Office data.

Figure 2-1. Number of Software-Related Patents Granted per Year by USPTO, 1991 to 2011

Some have commented on the extent to which the U.S.A.'s stand on patenting computer programs has greatly influenced other jurisdictions. Recently, Ravindra Chingale, in the Oxford Journal of International Intellectual Property Law and Practice has written the following:

The decision of the US Supreme Court in Alice Corporation v CLS Bank International 573 US (2014) has significantly affected attitudes to software patenting worldwide.[136]

In determining what is patentable about software, U.S. courts have been struggling to establish

[136] Ravindra Chingale, ''Alice and software patents: implications for India", (2015), 10 J Intell Prop L & Prac. 5 at 353

tests for many years. Hence, the rise and fall of the patent as a protection mechanism for computer software innovations in the U.S. has been witnessed in the last six decades. The evolution of software patents began with three Supreme Court cases as the technology was evolving from the mainframes into the PC era.[137]

Most information technology firms such as IBM,[138] Samsung, Canon, Panasonic, Toshiba and Microsoft are being awarded patents by the U.S. Patent and Trade Mark Office (USPTO)[139]. Issues of patenting computer programs date back to the 1972 case of *Gottschalk v Benson*[140]. In *Benson* case the court asked whether the claim would wholly preempt a mathematical algorithm. This is one test. Then, the court said, *"The patent would wholly pre-empt the mathematical formula and in practical effect would be a patent on the algorithm itself."* However, the court gives a very restrictive meaning[141] to the term algorithm- Procedure for solving a given type of mathematical problem".[142]

In *Parker v. Flook*[143], the court asked whether the claim process contributed to the article's transformation in state or nature. At first, the patent examiner rejected[144] the claim, arguing that the only novel invention in this claim was the mathematical formula. Similarly, the Patent Trial and Appeal Board affirmed the examiner's rejection. However, the Court of Customs and Patents Appeal (CCPA) granted the patent by reversing the decision of the board and examiner.[145] Nonetheless , the Supreme Court finally reversed the decision of the CCPA, explaining: "*Respondent's process is unpatentable under § 101 not because it contains a mathematical*

[137] There was, however, one other software patent granted by the United States Patent and Trademark Office (USPTO) in 1968- U.S. Patent No.3, 380,029. For further information, see the discussion by Gene Quinn, "The history of software patents in the United States" IPWatchdog (03 October 2014), online: Patent bar Review <http://www.ipwatchdog.com/2014/11/30/the-history-of-software-patents-in-the-united-states/id=52256/>
[138] In 2002 , IBM alone was issued 3411 patents, most of them relates to software, See Arun Mehta, "The Absurdity of Software Patents",(11 December 2003) *http://world-information.org/wio/readme/992006691/1078487756*
[139] Supra note 30 (John a. Gibby) at 16
[140]*Gottschalk v. Benson*, 409 U.S. 63, and the Supreme Court in this case developed a machine-transformation test. The court there said the transformation and reduction of an article to a different state or thing' is the clue to the patentability of a process claim that does not include particular machines. Under that test, a computer program is patentable if and only if "(i) it is tied to a particular machine or apparatus, or (ii) it transforms a particular article into a different state or thing.
[141] *Supra* note 36 at 305.
[142] *Supra* note 143 at 65.
[143] Parker v. Flook, 437 U.S. 584 (1978).
[144] *Ibid*, Flook, at 588.
[145] This court is now replaced by the Federal Circuit courts.

algorithm as one component, but because, once that algorithm is assumed to be within the prior art, the application, considered as a whole, contains no patentable invention."[146]

Three years later, in *Diamond v. Diehr*[147] the court again reconsidered its decision and ruled that computer program can be patented.[148] In this particular case, the test used by court was whether the claimed process involves the transformation of an article, transforming uncured synthetic rubber into a different state or thing.

Then the court, in finding the computerized process patentable, explained:

"A claim drawn to subject matter otherwise statutory does not become nonstatutory simply because it uses a mathematical formula, computer program or digital computer. . . . A process is not unpatentable simply because it contains a law of nature or a mathematical algorithm. It is now commonplace that an application of a law of nature or mathematical formula to a known structure or process may well be deserving of patent protection. As Justice Stone explained four decades ago: "While a scientific truth or the mathematical expression of it, is not a patentable invention, a novel and useful structure created with the aid of knowledge of scientific truth may be"... Arrhenius' equation is not patentable in isolation, but when a process for curing rubber is devised which incorporates in it a more efficient solution of the equation, that process is at the very least not barred at the threshold by Section 101.[149]

Before the Supreme Court entertained in 2010 other software related patents, the Federal Circuit was struggling with determining the patentability of software claims. In consequence, the Court adopted different tests. The *Freeman-Walter-Abele Test*[150] is one often-cited test the Court has applied. This test was based on the concept of preemption, and attempted to distinguish claims that wholly preempt mathematical algorithm from those that did not. The focus of this was on

---

[146] *437 U.S. 584 (1978)*, supra note 140.
[147] *Diamond v. Diehr, 450 U.S. 175 (198).*
[148] *Ibid*, in this case it has been said, in the realm of computer programs, the distinction between what is patentable and what is unpatentable lies in whether a computer program is an *application* of an abstract idea, which may be patentable, or instead an abstract idea itself, which is not.
[149] *Ibid; see also* Cathy E. Crtsinger, "Patent: Patentability: Computer Software,: AT&T Corp. v. Excel Communications, Inc.", (2000) 15: 1 Berkeley Tech LJ, at 166 [In analyzing the patentability of these claims, the Supreme Court has consistently stated that, while a mathematical algorithm standing alone is an unpatentable abstract idea, a useful process that incorporates an algorithm may be patentable subject matter]; supra note 140.
[150] It the test developed out of series of three court decision called *Freeman, 573 F.2d 1237 (C.C.P.A. 1978); Walter, 618 F.2d 758 (C.C.P.A. 1980); and Abele, 684 F.2d 902 (C.C.P.A. 1982)*

patenting mathematical algorithm. In determining the patentability of claims, the Court identified two sub-step tests: whether the claim recites mathematical algorithm, and whether the claim as a whole is no more than the algorithm itself. If our answer is positive, then the claim is non-statutory subject matter.

The *Freeman-Walter-Abele test* was soon overridden by Federal Circuit.[151] . Hence, a decade of chaos and confusion followed as courts attempted to apply the test. During this era, the outcome of cases largely depended upon the particular Federal Circuit panel. Three camps of thought have been reflected among the Federal Circuit judges.[152]The first camp focuses on the preemption of all sorts of algorithm under the *Freeman-Walter-Abele test*. The radicals, on the other hand, believed in the patentability of software claims as long as the claim invention show some ''technical application and provides some technologically useful effect''.[153]Thirdly, some judges believed software claims could be patented if the claim relates to a machine.

Arrhythmia Research Technology Inc. v. Corazonix Corp[154] was the other software related case entertained by U.S Courts. The software is used in monitoring heart attack victims. Although there is a machine accepting input signals from the heart that is being monitored, the main invention is the software.[155] The Federal circuit granted a patent on this software invention by reversing the United States District Court for the Northern District of Texas decision.

In 1994, the Federal Circuit Court of Appeal again attempted to clear up some of the confusions in the *Alappat* case.[156] Kuriappan Alappat was granted a software patent.[157] In this later case, the focus shifted from *Freeman-Walter-Abele test* of preemption to *useful, concrete and tangible test*. Uncertainty continued for more years until the Federal circuit again developed another test of patentability. In 1998, the Court in the *State Street Bank case*[158] held that the transformation of data representing dollar amount by a machine to a series of mathematical calculation into the

---

[151] Emily Michiko Morris, "What Is "Technology"?", online: (2014) B.U. SCI. & TECH. L. (2014) at 30 http://fstp-expert-system.typepad.com/files/92-e.-morris_what-is-technology_iu_i.n..pdf .
[152] Supra note 86.
[153] *Ibid.*
[154] *Arrhythmia Research Technology Inc. v. Corazonix Corp* , 958 F.2d 1053, 22 USPO2d 1033 (1992)
[155] Supra note 143.
[156] 33 F.3d 1526. Also, see C. Mark Kittredget, "The Federal Circuit and Non-patentable Subject Matter Under In Re Alappat and in Re Warmerdam", (1995) 11 Santa Clara computer & High Tech. L.J. 261
[157] Patent no. 5,440,676;supra note 16.
[158] *State Street Bank & Trust Co. v. Signature Financial Group, Inc.*,149 F.3d 1368, 47 U.S.P.Q.2d (BNA) 1596 (Fed. Cir. 1998), cert. denied, 119S. Ct. 851 (1999)

final share price constitutes a patentable application of machine algorithm. This is because it produces a useful, concrete and tangible result in the form of a final share price.

Following these Court decisions, the U.S. patent office issued many patents (some referred them as ‘‘weak patents’’[159]), resulting in patent trolls, otherwise called non-practicing patents. Some, like David Hayes, attribute this to two reasons.[160] The absence of adequate database of prior arts of software in the USPTO is one reason. Secondly, the prevalence of aggressive practitioners who were seeking to protect software methods related to technology, in the decade of the internet is the other reason for the issuance of many software patents.

In 2008, the Court rejected the *Freeman-Walter- Abele test* and the *useful, concrete and tangible* result test adopted in *Alappat,*[161] *and state street bank test.*[162] Accordingly, the Federal Circuit in *Bilski*[163]adopted the machine-transformation test. If the claimed machine/process ties in with a particular apparatus or transforms a particular article to a different state or thing, then the claim is patentable.

The Supreme Court, in the recent *Bilski* case[164], has again rejected the machine-transformation test as a sole test of process patent eligibility[165]. The Court stated the ultimate determination must be whether the subject matter is a law of nature, physical phenomena or abstract idea, positing that these categories of subject matter are absolutely not patentable. As will be discussed below, the same Court analyzed the section 101 exception in *Alice*.[166] Finally, the court held the claims

---

[159] For more information about week and non-practicing patents, see Brian T. Yeh, *"*An Overview of the "Patent Trolls" Debate, Prepared for Members and Committees of Congress", CRS Report for Congress (16 April 2013) online: < https://archive.org/details/R42668AnOverviewofthePatentTrollsDebate-crs> ; and Anton, James J., Hillary Greene, and Dennis Yao, "Policy Implications of Weak Patent Rights", (2006) 6 Harvard Business school Innovation Policy and the Economy at 1–26.

[160]*Supra note 86.*

[161] Haewon Chung, "Lessons from Bilski", online: (2011) 9 CJLT 1 179 at 184 < https://ojs.library.dal.ca/CJLT/article/view/4846%3E>.

[162] Supra note 162,1373.

[163] *Bernard L. BILSKI Rand A. Warsaw* No. 2007-1130., 545 F.3d 943

[164] *Bilski v. Kappos,* 130 S. Ct. 3218, 3225 (2010)

[165] The court explicitly rejected the court of appeals stand, as the latter ruled that the machine-or-transformation test, was the sole test to be used for determining the patentability of a "process" under the Patent Act, 35 U. S. C. §101. See the analysis of Justice Kennedy in *Bilski* (2010) ; Michael B., Abramowicz, James E. Daily, and F. Scoot Kieff, *Perspective on Patentable Subject Matter*, Cambridge (2015), p-33

[166] *Alice Corp. v. CLS Bank Int'l*, 134 S. Ct. at 2354, [Section 101 "contains an important implicit exception: laws of nature, natural phenomena, and abstract ideas are not patentable."]

unpatentable in that case because they were directed to the abstract idea of hedging risk.

In 2015, the Supreme Court in *Alice Corporation Pty. Ltd. v. CLS Bank International et al*[167], again established another test of patentability: a two-step test. In arriving at this test the Court used its 2012 *Mayo v. Prometheus* case[168]. First, the [Court should] determine if "the claims at issue are directed to one of those patent-ineligible concepts"[169]. *Secondly,* "If so, the [Court should] then ask, what else is there in the claims before us?"[170] In the latter step, the Court is asking if there is an *inventive concept* that amounts significantly more than the patent ineligible concept itself. The Court explained this in the following manner:

> We have described step two of this analysis as a search for an "inventive concept—i.e., *an element or combination of elements that is 'sufficient to ensure that the patent in practice amounts to significantly more than a patent upon the [ineligible concept] itself.* "[171]

The *Alice* Court, using the two step tests, tried to distinguish patents that claim patent ineligible subject matters from patent eligible ones. It, then, held- implementation of wholly computer generated elements is not sufficient to add something to save the claims. The system claims to recite the *abstract idea* of implementing a generic computer. It then concluded that "the method claims, which merely require generic computer implementation, fail to transform that abstract idea into a patent-eligible invention."[172] Simply stated, *Alice Corporation's* innovative idea i.e. "concept of hedging" or "settlement of risk" is found a patent ineligible abstract idea. Furthermore, the Court stated *Alice's* claim does not add an element that transforms the patent ineligible abstract idea. Accordingly, the Court held *Alice's* registered patents were invalid as they fell under §101 exception.[173] The Court, first looked *Alice's* if it is directed to patent-ineligible abstract idea. The answer was positive. It then applied the second test i.e. if the claim

---

[167] Ibid. *Alice Corp*. claims a patent on a method of intermediating financial settlement using a computer system. Respondents, on the other hand, argued against the patentability of those claims based on §101of the U.S. Patent Act.
[168] *Mayo Collaborative Services, Dba Mayo Medical Laboratories, Et Al. V. Prometheus Laboratories, Inc.,*132 S. Ct. 1289 (2012)
[169] Supra note 171 at 2355
[170] Ibid
[171]Ibid
[172] Ibid at 2357
[173] Amanda Liverzani, "Fate of Software Patents Still Unclear Following SCOTUS Decision in Alice v. CLS Bank", Harv JL & Tech (28 June 2014), online: Harvard Journal of Law & Technology Digest <*http://jolt.law.harvard.edu/digest/fate-of-software-patents-still-unclear-following-scotus-decision-in-alice-v-cls-bank*> .

has an inventive step which would help it pass the § 101 exception. This time the answer was negative; hence, it did not pass the two-step tests.

Further controversy continued and new tests of patentability have been introduced by Courts. *Ultramercial Inc. v. Hulu LLC (Fed. Cir. 2014)*[174] is another patent case handled by Courts post-*Alice.* In this case *Ultramercial, Inc*. sued Hulu, YouTube and WildTangent for patent infringement. The claimant had its patent registered in 2008.[175] The defendants moved to make the claimed patent invalid. The District Court decided in their favor and dismissed *Ultramercial's* claim. The Court used the machine-transformation test and abstract idea[176] exception, mentioned above, in rejecting the claim. Later, the Federal Circuit rejected the District Court's decision and introduced two other tests in assessing the patentability of computer programs.[177] These tests are 1) the requirement of complex programing and 2) the use of the programs in the internet and cyber market environment or electronic commerce (electronic commerce over the World Wide Web).[178]

In summary, the U.S. *Patent Act* does not exclude the patentability of software technologies. All levels of Courts are developing different criteria of patenting software since the early days of 1970s. Likewise, the USPTO is struggling in entertaining software patent claims. The office has also issued thousands of software patents.

2.1.2 Software patenting in Canada

The approach taken as regards patenting software varies across jurisdictions. In some jurisdiction such as the U.S., we see leniency in permitting patents for software. The early U.S. Supreme Court case of *Diehr*[179] and some of its subsequent decisions support this benevolence in granting

---

[174] *Ultramercial, LLC v. Hulu, LLC,* 657 f.3d 1323 (Fed. Cir. 2011), reh'g and reh'g en banc denied, No. 2010-1544, 2011 U.S. App. LEXIS 25055 (Fed. Cir. Nov. 18, 2011).

[175] U.S. Patent No.7,346,545

[176] The Harvard Law Review Association, "Patent Law - Patentable Subject Matter - Federal Circuit Applies New Factors in Deciding Patentability of a Computer Program. - Ultramercial, LLC v. Hulu, LLC', 657 F.3d 1323 (Fed. Cir. 2011), reh'g and reh'g en banc denied, No. 2010-1544, 2011 U.S. App. LEXIS 25055 (Fed. Cir. Nov. 18, 2011)," online: (2012) 125 Harv. L. Rev. 2167 at 2169 <http://www.jstor.org/stable/23214434?seq=1#page_scan_tab_contents>

[177] Ibid at 2170

[178] *Ibid* at 2168

[179] *Supra* note 151.

patents. On the contrary, the EU approach as will be discussed below is a bit different. This section examines the Canadian approach.

In a statutory regime similar to the practice in other countries, the grant and administration of patent in Canada is guided by the *1985 Canadian Patent Act.*[180] In Canada, the *Patent Act* was interpreted as excluding computer programs and algorithms as non-statutory subject matter.[181] Section 27(8) of the *Act* excludes certain subject matters and states, *"no patent shall be granted for any mere scientific principle or abstract theorem. "*[182] Some argue that computer program might fall under the abstract theorem exclusion.[183]The reason for this is that computer software involves algorithms, and the latter are regarded as abstract theorems.[184] But such interpretation will only consider "*computer programs per se"*, or "*computer programs as such",* to use the European terminology. David Vaver states that before 2005, the Canadian Intellectual Property Office (CIPO) considered computer programs as unpatentable subject matter for the reason that they would halt the emerging field.[185] The *Patent Act* being one major source of law regarding patentability, the patent office has adopted numerous supplementary Notices and guidelines. The 2007 CIPO's manual seems to mitigate its pre-existing position. According to this manual, computer programs could be amenable to patentability provided they are ''integrated with traditionally patentable subject matter''[186].

Conrad Delbert Seaman, in his recent article, has properly articulated the *current* position of Canadian software patents.[187] The author put the Canadian approach as falling between the U.S.

---

[180] *Canadian Patent Act*, R.S.C. 1985, c P-4

[181] Ibid, See *also Schlumberger Canada Ltd. V Commissioner of Patents , 56, 204(1984*), see also Eloise Gratton, "Should Patent protection be Considered for Computer Software- related Innovations?", (2003) VII Computer L Rev & TJ, at 225-226

[182] *Tennessee Eastman Co. v Canada (Commissioner of Patents*) (1972), 8 CPR (2d) 202 (SCC), P- 204.

[183] Conrad D. Seaman, "Contextualizing the Software Patent Debate in Canada: A Practical Approach to Policy Development", (2014) 3:1 97 Osgoode Hall Review of Law and Policy 3.1 97 at 103; *supra* note 156 Eloise Gratton, at 225.

[184] *Ibid*, Eloise Gratton.

[185] Supra note 187 at 105; David Vaver, *Essentials of Canadian Law: Intellectual Property Law: Copyright, Patents, Trademarks* (Concorde Ontario: Irwin Law Concorde Ontario, 1997) at 129.

[186] Canadian Intellectual Property Office - Manual of Patent Office Practice", March 2007 at c.12 and c.16 as quoted by Seaman, Conrad Delbert, p-105.

[187] Supra note 187.

and Europe, and described it as "a non-position"[188]. The *Patent Act* does not mention computer programs at all, either as an exclusion or patentable subject matter. The "non-position" claim seems to arise from the lack of clarity on the part of the judiciary and the Patent Office.

In recent years, though, the Canadian Patent Office has eased its restrictions on patenting computer-related inventions. Patents are now rather routinely granted for inventions in the computer and information processing field.[189] Even the CIPO amended guideline once considered computer programs as patentable subject matter.[190] A case in point is the recent patent granted in *Amazon.com Inc*.[191] This case involves a method claim whereby a customer's profile data will be saved in their own computer. Additionally, the method saves a user's identification information in the customer's server computer. The method is called "one click" buying. It allows users to transact in online marketplaces using the predefined profile, mailing and payment information. In this case, although the Patent Commissioner rejected Amazon's request based on *Schlumberger Canada Ltd.*[192], the Federal Court of Appeal reversed its decision.

Following the *Amazon* case, Patent Notice Practice guidance for examiners of computer implemented innovations was prepared in March 2013.[193] Based on this latest notice, although computer-implemented inventions may be claimed as a *method, machine, product,* computer arts – including computer programs – may not be claimed *as such.*[194]This is in accordance with the earlier manual. The 2007 manual allows the patentability of computer programs so long as the claim is integrated with other patent-eligible subject matter. Few additional facts have been included in the 2013 Notice-evaluation of computer program's patentability according to section 2 of the *Patent Act* should adhere to purposive construction. The other most important additions

[188] *Ibid* at 100,105.
[189] *Ibid* at 226.
[190] *Ibid*, at105; Canadian Intellectual Property Office - Manual of Patent Office Practice", March 2007 at c.12 and c.16 (revised in 2009). [This manual, in section 12.06.02, covers computer programs. Accordingly, if the program is essentially abstract in character it is not an invention. However, as explained below, it could be amenable to patent protection if it meets the technical contribution criteria].
[191] *Canada (Attorney General) v. Amazon.com, Inc.,* [2011] FCA 127.
[192] Supra note 185 (*Schlumberger Canada Ltd v Canada (Commissioner of Patents).*
[193] See, Canadian Intellectual Property Office, Examination Practice Respecting Computer-Implemented Inventions, PN 2013-03
[194] Ibid

in the manual, explains the convergence of the Canadian approach to the European Patent Office's and some of the U.S.'s patentability tests: the technical solution to technical problem approach. A computer program could be patentable if it meets Section 15.05.03 of this manual. The manual requires for claims to provide a novel and unobvious technological solution to a technological problem. The presentation of contribution is not enough. It rather should provide technological solution to a technological problem.

2.1.3 Software patenting in the European Union

Generally, computer programs, as such, are excluded from patentability in the EU.[195] But we see European authorities mitigating this exclusionary clause with the fulfillment of one requirement: if the program has a technical effect, software related inventions[196] can be patentable.

A relevant law for the analysis of patentability of computer programs in EU is the European Patent Convention (EPC).[197] Our analysis should begin with a discussion of Article 52 of this convention. The first Subsection of this provision sets out the requirement of patentability. Accordingly, an invention would be patentable if it meets three cumulative requirements: industrial application[198], novelty[199], and inventive step.[200] On the other hand, Sub-Article 2 lists excluded subject matters. Programs for computers are among the excluded subject matters.[201] What is excluded from the realm of patent is a pure computer program, in the abstract[202] as Sub-

[195] Article 52(3) of EPC (see *below* 172)
[196] ''Software patents'', ''software related inventions'' and ''computer-implemented inventions'' are used interchangeably by bulk of literature and case laws. See, Philip Leith, *Software and Patents in Europe, Cambridge Intellectual Property and Information Law*, (Cambridge University Press), (2007), p-16 [indicating, at times these terms can frequently be used interchangeably with ''business method patents'']; The proposed directive on the patentability of computer implemented inventions (CII) defines CII as ''*any invention the performance of which involves the use of a computer, computer network or other programmable apparatus and having one or more prima facie novel features which are realized wholly or partly by means of a computer program or computer programs''(*see article 2 (a) of Proposal for a Directive of the European Parliament and of the Council on the Patentability of Computer-Implemented Inventions, COM(02)92 final available at http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM:2002:0092:FIN.
[197] Convention on the Grant of European Patents (European Patent Convention of 5 October 1973 as revised by the Act revising article 63 EPC of 17 December 1991 and the Act revising the EPC of 29 November 2000
[198] Ibid, article 52 (1) & 56 [*An invention shall be considered as susceptible of industrial application if it can be made or used in any kind of industry, including agriculture]*
[199] Ibid, *an invention shall be considered to be new if it does not form part of the state of the art, see article 52(1) &54.*
[200] *Ibid,Article 52(1), 56 [it is not obvious to a person skilled in the art]*
[201] *Ibid* article 52(2(c))
[202] Paul England, ''Computer-related inventions: from CFPH to Macrossan", (2007), 2 J Intell Prop L & Prac. 5 305 at 306.

Article 3 reads: Paragraph 2 shall exclude the patentability of the subject-matter or activities referred to therein only to the extent to which a European patent application or European patent relates to such subject-matter or activities as such.

The 1985 EPO guideline for examination, as amended in September 2016, has also addressed excluded subject matters. Chapter VIII of the latest version deals with excluded subject matters. Section 2.2 of this chapter is particularly concerned with sections 52 (2) & (3) of the EPC. The guideline classifies those subject matters into two. The first limbs are non-technical. The guideline, by this limb, wants to address excluded subject matters *as such*. In other words, article 52(3) matters are regarded as non-technical.[203] Article 52 (2) of EPC lists excluded subject matters, and Sub-Article (C) of this convention includes programs for computer. In other words, the guideline considers, pure computer programs mentioned by article 52 (3) of the convention as non-technical subject matters, and not patentable.

The second limb concerns claims involving technical features. These matters are listed under Article 52 (2) of the convention. Though the convention states those matters are not patentable inventions, the guideline qualifies the convention. Accordingly, if these matters demonstrate technical features and contribution, they could be treated as patentable inventions. Significant weight seems to be given to the "contribution" element. On the other hand, the claims contribution may serve a technical purpose though it appears to be non-technical in its feature.[204] The EPO guideline adopts the "technical solution to technical problem" criterion which is incorporated in the Canadian Patent Office examination practice regarding computer implemented inventions.[205]

Be this as it may, we see variations in approaches between member states and the European Patent Office. For instance, the UK's patent office and Courts used to follow the 'technical contribution' approach, whereas the EPO considers whether the claim has a 'technical feature at

[203] EPO, Guidelines for Examination in the European Patent Office, Nov 2016, ISBN 978-3-89605-158-5< http://www.epo.org/law-practice/legal-texts/guidelines.html >.
[204] *Ibid*, section 2.2
[205]*Supra* note 194.

all'.[206] Stefan Steinbrenner, former chairman of an EPO Technical Board of Appeal, said the following.

> "Any of the subject-matters listed in Article 52(2) EPC may comprise an invention if it has technical character or contributes to it (in particular because a technical problem is solved by using technical means or a technical effect is achieved, technical interactions occur or technical adaptations are effected, in other words: if such subject-matter lends itself to a technical application*."*[207]

This shows that the patentability of computer programs should be assessed on a case by case basis. As can be noted from the above remark, the technical element is repeatedly used: technical problem, means, effect, adaptation, contribution, interaction, character and application. This may be the reason for EPO granting thousands of software patents. It is not only the U.S. authorities who are generous in granting software patents. The EPO, in its 1994 annual report, noted that about 11,000 software patents have been granted.[208] A data from 2007 show the issuance of 8,981 patents classed under computing.[209]Since 1978, more than 30,000 software related patents have been issued by European Patent Office.[210] Andrés G. González, in his interview for WIPO magazine, concurred with the idea of the European Patent Office issuing more than 30,000 software patents.[211]

Even so, there are researchers who question the requirements of patentability of computer programs. This is because extensive case law in the field shows that the 'technical requirement', the main pillar of the traditional European patent system, as applied to computer programs, has

---

[206] *Supra* note 208. Notice however, should been taken that this technical contribution and technical feature analysis is used only to identify whether the subject matter is excluded or not. The other requirements of patentability set forth under sub-article 1 remains intact

[207] Stefan Steinbrenner, "The patentability of computer-implemented inventions", EPO (24 March 2011) <http://archive.is/e-courses.epo.org>.

[208] Philip Leith, *Software and Patents in Europe,* (Cambridge, UK: Cambridge University Press, 2007) at 16

[209] *Supra* note 137.

[210] Robert Bray, *The European Union "Software Patents" Directive: What is it? Why is it? Where are we now?,* (2005) Duke L & Tech Rev11 1-18.

[211] David Koepsell, *Innovation and Nanotechnology: Converging Technologies and the End of Intellectual Property,* (New York, U.S.: Bloomsbury Academic, 2011) at 17 [analyses the increasing patenting of software, and he went on further in elaborating the fact that wealthier software companies have strong patent portfolios]; *Supra* note 137.

repeatedly proven inappropriate and confusing.[212]

The commission tried to make changes regarding patenting computer programs. Accordingly, it sought suggestions from the general public, interest groups and member states. To achieve this purpose, it announced a consultation in 2002.[213] The bulk of responses to the consultation came from a petition for a patent-free Europe organized by *EuroLinux*, and the results of the consultation indicated that ninety-one percent of the respondents opposed software patents.[214] In 2005, the European parliament rejected its proposed directive. However, the debate over the patentability of software remains in Europe to this day.[215] So the decision to withdraw the proposed directive does not mean that the issues addressed in it and the interests affected by it have been resolved. Real debate has merely been deferred, and it is important to recognize the issues and interests clearly before the debate is resumed.[216]

The United Kingdom, observed at the national level, has an interesting approach toward software patenting.[217] In *Slee & Harris application (1966)* RPC 194[218], the examiner granted a patent for a program directed to a machine. In the same case, a separate claim of patenting the program itself was requested, and the examiner allowed the claimed patent: ‘‘Linear programming means for use in controlling data processing apparatus.’’[219]

The other early English software case for which a patent was granted was the *International Business Machines Corporation's application [*1980] *FSR 59[220]*. The Patent Appeal Tribunal agreed with the superintending examiner's view in allowing the patent, and explained as follows:

…what Mr. Nymeyer seeks to claim as a manner of new manufacture is a method involving operating or controlling a computer in which the computer is programmed in a particular way or

[212] Rosa Maria Ballardini, ‘‘Software patents in Europe: the technical requirement dilemma", (2008), 3 Journal Intell Prop L & Prac 9 563.
[213] *Supra* note 20. See also *supra* note at 229.
[214] *Ibid*; *supra* note 185.
[215] John R. Allison, Abe Dunn & Ronald J. Mann, "Software Patents, Incumbents, and Entry", (2006-2007) 85 Tex. L. Rev. 1579 at 1621.
[216] Andres Guadamuz Gonza´lez, ‘‘The software patent debate’’, (2006) 1 Journal Intell Prop L & Pract 3 at 196
[217] Despite s. 1(2) of the 1977 *UK Patent Act* exclude the patentability of pure software inventions, the Intellectual Property Office (IPO) and Courts have allowed software patents.
[218] *Supra* note 21
[219] *Ibid.*
[220] *Ibid* at 129.

programs in physical form to control a computer so that it will operate in accordance with his method. The method is embodied in the program and in the apparatus in physical form and in our view the claims should be allowed to proceed [*patentable*]. We agree with the superintending examiner that the law is that an inventive concept, if novel, can be patented to the extent that the claims can be framed directed to an embodiment of the concept in some apparatus or process of manufacture.

When we see the above two cases (three claims), it seems the main emphasis is the embodiment of programs to physical medium. In the first claim of *Slee and Harris* case, the examiner reasoned " …the claim is directed to a machine which has been…", and in the '*Linear programming'* claim of the same case, the examiner opined that as " …the means claimed is an *integer which physically cooperates with a computer*…..therefore, *when fixed in a machine*…". The same holds true in the IBM case.

2.2 Copyrighting Computer Software

When the issue of computer program protection came up in the 1970s and early 80s there was a fair deal of uncertainty about how to deal with these programs under copyright law. The question then was whether the existing rules on copyright law could apply to computer programs without amendment or further refinements.

Generally, the line was taken that computer programs can fit reasonably comfortably under the category of literary works.[221] So, copyright protection is the commonly accepted method of protecting computer programs. Accordingly, since the 1980s, in many countries copyright in protecting computer programs is taken for granted.[222]It falls under the category of literary works, as the developer writes software instruction as other authors of literary works do.

There were, however, questions about the degree of comfort copyright gives to authors of computer programs.[223] This is because, unlike other works protected by copyright, the nature of

[221] The provisions of WCT and TRIPS agreement with their referral sections to the Berne Convention on the Protection of Artistic and Literary works is the manifestation of this fact.
[222] Deborah Azar, "A Method to Protect Computer Programs: The Integration of Copyright, Trade Secrets, and Anticircumvention Measures" online: (2008) Utah Law Review 4 1395 at 1397 http://epubs.utah.edu/index.php/ulr/article/view/135/117 .
[223] See, for instance, *Raymond T. Nimmer & Patricia Ann Krauthaus, "*Software Copyright: Sliding Scales and Abstracted Expression", (1995) 32 Hous. L. Rev. 317 ; There were also litigations regarding the copyrightability of

software is unique. Generally speaking, computer software is technical in its attributes.

2.2.1 Copyrighting software: International instruments

As stated above, the U.S., Canada, and the EU choose copyright as the best method of protection. At the international level, we have the World Intellectual Property Organization Copyright Treaty (WCT) and Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS) with referral provisions to the *Berne Convention*. These are the most recent international copyright instruments governing copyright at the international level. The non-inclusion of computer software in the earlier laws could be chalked up to many reasons. Problems related to prediction software inventions[224] in that time is one reason. The other one is global leaders of the field such as the U.S.[225] did not want to put the issue in the negotiation process.

2.2.2 Copyrighting software in the U.S

At the national level, too, most jurisdictions choose copyright to protect computer software. However, there were some objections regarding considering computer programs as copyrightable subject matter. In one case from Australia,[226] a Trial Court held that none of the programs were literary works within the meaning of 1968 *Copyright Act*,[227] but this case was reversed on appeal. Professor Samuelsson and her colleagues argue copyright offers very thin protection against software copying.[228] Although we have these and related critical comments against the copyright protection of computer software, the fact is that copyright remains, at least legislatively speaking, a main method of software protection in the U.S.

It was in 1964 that the U.S copyright office registered two computer programs for the first time.[229]Be this as it may, the first legislative initiative in determining the *scope* of copyright for

---

computer computers in the U.S.A and Australia, see for instance *Computer Edge Pty. Ltd.* v. *Apple Computer Inc.* (1986) 161 CLR 171.

[224] Nick Bassil, ''An introduction to international IP instruments relevant to electronics and software'' in Nicholas Fox, Sian O'Neill & Carolyn Boyle, eds, *Intellectual property in Electronics and Software: A Global Guide to Rights and Their Applications (London: Globe business Publishing Ltd. 2013)* at 15

[225] Orrin G. Hatch, ''Better Late Than Never: Implementation of the 1886 Berne Convention'', (1989) 22: 2 Cornell Inter'l LJ at 4.

[226] Ibid; see also Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240 (3rd Cir. 1983).

[227]J.McKeough, ''Apple Computer Inc. V. Computer Edge Pty Ltd", A Case Note'', (1984) UNSWLJ 162 at 164

[228] *Supra* note 29 (Robert, at 281).

[229] Lee A. Hollaar, *Legal Protection of Digital Information*, (Washington DC, USA: BNA Books, 2002) at 57

computer software in the U.S came into the picture in 1980[230] when Congress amended the *1976 Copyright Act.*[231] The intention of Congress to extend copyright protection for software was explicit in the *1976 Copyright Act*.[232]To further investigate the copyright issues on computer programs, Congress allowed additional time to the National Commission on New Technological Users of Copyrighted Works commonly referred to as CONTU.[233] The Commission finally proffered two recommendations. Its first recommendation is the inclusion of a definition in the *Copyright Act.*[234]This recommendation was accepted and computer program was defined in the s. 101 of the *1980 Copyright Act*[235]. Secondly, it recommended amendment of the limitation clause, § 117 of the *Act.*[236] It allowed the owner to make copies and adaptations as long as they are used for archival purposes.

It was not only the legislative initiatives that attracted attention. Parallel judicial developments, though tortuous, were underway. In 1982, the Federal Circuit handed down its decision in *Williams Electronics, Inc. v. Artic International, Inc.*[237]The Court affirmed copyright protection of computer programs and clarified traditional scope of software copyright, i.e. object and source code. Furthermore, it addressed the fixation requirement mentioned in § 101. The Court of appeal has also entertained for the first time the software copyright issue in the case of *Apple Computer, Inc. v. Franklin Computer Corp.*[238] Apple Computer Inc. brought the case to the District court in 1982 alleging copyright infringement, among other things. The District Court denied Apple's motion and the plaintiff lodged an appeal. The defendant raised four defenses of which three are important for our consideration. Firstly, it argued that machine readable codes are not copyrightable. Secondly, the defendant challenged programs stored in the internal memory of a computer (ROM). Thirdly, it regarded operating system software as an *idea* rather

[230] Karen J. Kramer, "Extending Copyright Protection to a Computer Program's Structure. Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc. 797 F.2d 1222 (3d Cir. 1986)", online: (1987) 65:2 Wash. U. L. Q. 471 at 474 < http://openscholarship.wustl.edu/law_lawreview/vol65/iss2/6/> .
[231] *Copyright Act* , 17 U.S.C. (1976).
[232] Supra note 233, at 58.
[233] Public Law 93-573 and Public Law 95-146 [The commission (a composition of fourteen expertises) has been established in 1967 to assist the president and Congress in adopting national copyright policy].
[234] Supra note 233 at 60.
[235] 17 § 101(1980).
[236] *Supra* note 233.
[237] *Williams Electronics, Inc. v. Artic International, Inc*, 685 F.2d 870 (3d Cir. 1982).
[238] *Apple Computer, Inc. v. Franklin Computer Corp, 714 F.2d 1240 (3d Cir.1983).*

than *expression of idea*. The Court of Appeal for the Third Circuit, by heavily relying on the CONTU report, reversed the motion and decided in favor of Apple.

The importance of copyrighting software began in parallel with computing evolution. Copyright did not protect software in the 1950s-mainframe era. This means the use of software in mini and personal computers (PC) is more widespread than in the small number of mainframe computers. The early commercialization of PC software began to expand when the first spreadsheet computer program known as VisiCalc[239] was released for Apple II in 1979. Then, Lotus development (later part of IBM) released its vital software, Lotus 1- 2- 3 in later years. With the further release of Mackintosh in 1984, the software market began to explode. The decade witnessed intense competition in the software business. We also saw a wavering uncertainty about software patents, particularly in the Supreme Court. This uncertainty about software patents and strong competition in the software industry seems one reason for software developers to push the confines of copyright protection. Software as a copyrightable subject matter used to be regarded as a literary work[240]; as such, copyright protected the literal element of it. As time passed, companies began to claim copyright protection for the non-literal element of software. This non-literal element of software is commonly referred to as the "look and feel" of computer software.[241] Non-literal aspects of software include visible and invisible aspects of software. The move to copyright non-literal aspects of software is, in the words of Pamela Samuelsson, the move against the "minimalist" or "thinner" copyright protection of software.[242]

Many "look and feel" copyright cases were filed and decided after the launch of Lotus 1-2-3; and the broader *scope* of copyright protection was fueled by the Federal Circuit's decision in *Whelan Assocs., Inc. v. Jaslow Dental Laboratory, Inc*.[243] The defendant developed a new program using

[239] Tom Hormby, "VisiCalc and the Rise of the Apple II" Apple History (25 September 2006), online: Low End Mac's Online Groups < http://lowendmac.com/2006/visicalc-and-the-rise-of-the-apple-ii/>.
[240] Appropriate copyrightable works, in all jurisdictions, relate to literary, artistic, musical and dramatic works. However, their proper scope is the subject of fierce judicial and academic discourse.
[241] See generally, Pamela Samuelson, "Why the look and feel of software user interfaces should not be protected by copyright law", online :(1989) 32Communications of the ACM 5 http://www.foo.be/andria/docs/p563-samuelson.pdf
[242] Pamela Samuelson, "Reflections on the State of American Software Copyright Law and the Perils of Teaching It", online: (1988) 13 Colum.-VLA J.L. & Arts 61 (1988) at 62 < http://scholarship.law.berkeley.edu/facpubs/128/>
[243] *Whelan Assocs. Inc. v. Jaslow Dental Laboratory Inc*., 797 F.2d 1222 (3d Cir. 1986) *cert. denied*, 479 U.S. 1031 (1987).

similar interfaces and different high-level programming language (BASIC). The first *Dentlab* software ran on IBM minicomputer; however, the later[244] ran on IBM PC. It was a copyright infringement case, and the disputable software was dental laboratory software. In this case, the court ruled that *"the line between idea and expression may be drawn with reference to the end sought to be achieved by the work in question. In other words, the purpose or function of the utilitarian work would be the work's idea, while everything that is not necessary to that purpose and function would be part of the expression of that idea."*[245] The main basis for the copyright infringement verdict, in this case, was interface similarity between the two programs. For Samuelsson, *Whelan* is the strongest expression of the maximalist view.[246] This case also established the *structure, sequence and organization* (SSO) test of software.[247]

Four years after *Whelan*, in *Lotus Dev. Corp. v. Paperback Software and Mosaic Software*[248] the District Court decided another landmark computer program and user interface case. Lotus development owned "*Lotus 1-2-3"* spreadsheet, and the defendants owned another spreadsheet known as "*VP Planner"*. Unlike *Apple Computer, Inc. v. Franklin Computer Corp, Lotus* concerned application program software. The plaintiff claimed Paperback Software and Mosaic Software's *"VP Planner"* software infringed its copyright over "*Lotus 1-2-3".* In contrast, the defendants argued against the copyrightability of the non-literal element of software. The court determined that Lotus had a copyright over the "look and feel" aspects of its user interface.[249]

In 1992, the 2nd circuit in *Computer Associates International Inc. v. Altai Inc*.[250] tried to articulate a more reliable and analytical framework for adjudicating software copyright cases. Computer Associates (CA) had a job-scheduling computer program, Adapter, for IBM mainframes. The defendant had a similar purpose program called "ZEKE". Altai hired one of the former employees of CA to write a new program, "OSCAR 3.4". While writing "OSCAR", the

---

[244] The first *Dentlab* program was written in EDL programming language. Later, Jaslow developed similar program using BASIC programming language.
[245] *Ibid*, at 1235-40.
[246] *Supra* note 246.
[247] The court developed this test to determine if one software infringes the copyrighted works of others. It is a test which helps determine the copyright violation of non-literal elements of software.
[248] *Whelan*, in *Lotus Dev. Corp. v. Paperback Software and Mosaic Software* , 740 F. Supp. 37 (D. Mass. 1990).
[249] Josh Lerner and Feng Zhu, "What is the impact of software patent shifts? Evidence from Lotus v. Borland", online: (2007) Int. J. Ind. Organ. 25 at 514 < http://www.nber.org/papers/w11168>.
[250] *Computer Associates International Inc. v. Altai Inc* , 75 F.Supp. 544, 20 USPQ2d 1641.

programmer copied portions of "Adapter's" source code, and CA claimed that the defendant infringed copyright by copying parts of its "Adapter". The defendant assigned a new team of programmers who did not have access to or information about Adapter, and developed "OSCAR 3.5". Determination was sought as to whether the "OSCAR 3.5" had infringed CA's "Adapter". The two most important issues framed by the court were: access and substantial similarity check. Although the court assumed the defendant's accessing of the plaintiff's Adapter, it concluded that the rewritten OSCAR 3.5 was not substantially similar to the Plaintiff's Adapter. The court established a so-called *abstraction, filtration and comparison* test. The test requires classifying works in different levels of abstraction, filtering out the protected elements in each level and comparing the remaining protected works. Hence, the Federal Circuit used this three-step test to determine the substantial similarity of the non- literal elements in *Computer Associates*' "Adapter" and *Alti*'s "OSCAR 3.5" software.

In parallel, the *Lotus Development Corp. v. Borland International, Inc.* [251] litigation was going on, which sought to determine whether computer menu command hierarchy was copyrightable. The District Court decided in favor of Lotus and regarded those menu command hierarchies as copyrightable. Borland appealed, and the Federal Circuit reversed the decision and decided in favor of the defendant. The Supreme Court decision was in a tie, and therefore, affirmed the Federal Circuit's decision.

Following the Supreme Court's decision, outcomes continued to oscillate, but the copyrightability of look and feel began to die. Be that as it may, copyright continued to be the main protection mechanism. For David Hayes, the increase in reliance on copyright as a software protection mechanism depends on four factors.[252] Firstly, copyright remained important for protection against piracy in the mass market, especially with the rise of peer to peer networking and mass downloading. The increase of uncertainty in software patents is the second factor. The third factor is the rise of open source software which requires copyright protection as a legal basis for enforcing the terms of an open source license, yet grants free and broad license rights. The final factor is the move of software into the cloud, which reduces the opportunity for piracy and creation of similar

[251] *Lotus Development Corp. v. Borland International, Inc* , 799 F. Supp. 203 (D. Mass. 1992) [Borland I1]; 788 F. Supp. 78 (D. Mass. 1992).
[252] *Supra* note 86.

programs.

In 2014, in *Oracle America, Inc. v. Google, Inc.*[253], the Federal Circuit declared that the Java APIS are copyrightable, potentially reinvigorating again the protection of functional and non-literal elements of software.

Oracle developed a software code called Application Programing Interfaces (APIs) for Java programming language. Google copied "declaring code," the "structure, sequence and organization" for 37 of the Java APIs. The case concerns whether APIs are copyrightable. The Court decided in favor Oracle, and said the "37 API packages—including the declaring code and the structure, sequence, and organizations are copyrightable".

2.2.3 Copyrighting software in Canada

Canadian laws and jurisprudence as regards computer program protection are not as well developed as in the EU and the United States systems. As discussed above, patent rules and judicial pronouncement pertaining to Canada are not clear. There is not even much study and academic discourse, again as compared to the EU and U.S. However, this does not mean there is no attempt to address the issues of computer software. In the U.S. the foundation or starting point in discussing IPRs protection is Article I, Section 8, Clause 8, of the United States Constitution. Similarly, the *Constitution Act*[254] of 1867 recognizes both patent and copyright protections. Hence, the *Copyright Act*[255] of Canada bases its source on this constitutional provision. Statutorily speaking, computer program as a copyrightable subject matter came on the scene in the 1988 copyright amendment *Act*.[256] This amendment, like the Unites States' CONTU, is the result of suggestion from the House of Commons Sub-Committee on the Revision of Copyright.[257] The report of the committee is commonly referred to as *A Charter of Rights for*

[253] *Oracle America, Inc. v. Google, Inc* , 750 F. 3d 1339 (2014).
[254] *Constitution Act,1867 (UK), 30 & 31 Vict, c 3, reprinted in RSC 1985, App II, No 5*[section 92(23) bestows subject matter jurisdiction to the Federal Parliament of Canada. Hence, copyright with its cousin intellectual property rights falls in the ambit of Federal Acts].
[255] Supra note 73.
[256] Supra note 58.
[257] Kimbery Hancock, "1997 Canadian Copyright Act Revisions", (1998) 13 Berkeley Tech. L.J. at 517. *http://scholarship.law.berkeley.edu/btlj/vol13/iss1/33/*>. Some considers one reason for the revision is to meet the U.S. standard , see, for instance, Peggy Berkowitz, "Canada Is Drafting New Copyright Law to Satisfy Grievances

Creators.[258] The revision process of the *Copyright Act* was implemented in two phases.[259]Phase one is particularly important for this discussion as it was a phase when computer program has acquired an explicit statutorily copyright protection.

Computer program has been considered as a literary[260] work even before the inclusion of the section 2 definition in the 1988 revised *Copyright Act*. Relevant cases in this regard are[261] *Spacefile Ltd v. Smart Computing Systems Ltd*[262], *IBM v. Spirales Computer Inc.*[263], *RDG Inc. v. Dynabec Ltd*[264]. The most important software copyright case before the revision of the *Copyright Act* was the *Apple Computer Inc. v. Mackintosh Computers Ltd Case*.[265] The *Apple* case is particularly interesting. This case involved three levels of courts (trial, appellate and Supreme Court). It involved the operating system copyright claim of Apple on "Applesoft" and "Autostart ROM". The defendant converted the written work of Apple to one of electrical code and encoded it on one of its chips. Madam Justice Reedfound the case in favor of the plaintiff, deciding that translating and reproducing (by encoding written programs on silicon chips) amounts to infringement based on section 3(1) of the *Copyright Act*. An appeal was lodged to the Federal Court of Appeal and the court dismissed the petition. A further petition was made to the Supreme Court and the court unanimously agreed with Madam Justice Reed's conclusion. Hence, this case

---

of U.S. Concerns", *Wall Street Journal* (29 April 1986) online: *Wall Street Journal* http://search.proquest.com/docview/398055666?rfr_id=info%3Axri%2Fsid%3Aprimo

[258] Standing Committee on Communications and Culture, Canada House of Commons, Report of the Sub-Committee on the Revision of Copyright, A Charter of Rights for Creators 4 (Issue No. 27, 1st Session, 33rd Parliament, 1985) as cited in Kimbery Hancock, "1997 Canadian Copyright Act Revisions", (1998) 13 Berkeley Tech. L.J. at 517; Vaver, David, Copyright Law: Recent Canadian Developments", Online: (1988) 16 *Australian Business Law Review at 413* <http://search.proquest.com/docview/223515078?OpenUrlRefId=info:xri/sid:primo&accountid=14739

[259] Jay Makarenko, "Copyright Law in Canada: An Introduction to the Canadian Copyright Act"Mapleleafweb (13 March 2009), Judicial System & Legal Issues < http://www.mapleleafweb.com/features/copyright-law-canada-introduction-canadian-copyright-act.html>.

[260] Barry S., Steven M., and Carys C., Copyright Cases and commentary on the Canadian and International Law, 2nd ed, 2013, Carswell, Canada,, ch 7 at 359.

[261] George E. Fisk & Jane E. Clark, "Hardware and Software Protection in Canada" online: (1990) X 10 Computer L.J. at 484-85 http://repository.jmls.edu/cgi/viewcontent.cgi?article=1424&context=jitpl

[262] *Spacefile Ltd v. Smart Computing Systems Ltd* , 75 C.P.R. (2d) 281 (1983).

[263] *IBM v. Spirales Computer Inc* , 80 C.P.R. (2d) 187 (1984).

[264]*RDG Inc. v. Dynabec Ltd*, 6 C.P.R. (3d) 299 (1985).

[265] *Apple Computer Inc. v. Mackintosh Computers Ltd* ,10 C.P.R. (3d) 1 (F.C.T.D. 1986); aff'd, 18 C.P.R. (3d) 119 (F.C.A. 1987).

for the first time made clear that computer program both in source and object was copyrightable. The case triggered Copyright amendment.[266]

The *Canadian Copyright Act*[267]*, in* section 2 provides a definitional clause for computer program. Section 2 of the *Canadian Copyright Act* is similar to section 101 of the *U.S Copyright Act.*[268] Interestingly, there is no common-law copyright in Canada, especially subject matter, and infringement issues[269].Accordingly, we only see very few software cases applied and entertained by the Canadian courts. Canada's membership in the WTO and UN (WIPO) and its commitment to the TRIPS agreement and the WCT is another important fact as to the copyrightability of computer programs in Canada. Hence, by the applying these two agreements Canada is duty bound to extend copyright protection to computer program.

2.2.4 Copyrighting software in the EU

As there is no separate law for computer programs in the U.S. or Canada, we simply apply the respective general copyright and patent laws in order to determine the nature of specific rights the right holder has. However, for the past 25 years, in the EU there was variation in approach as regards software protection.[270] Before adopting the software directive in 1991, member states of EU have regulated software differently. For instance, in some member states the degree to which software is required to be original to meet copyrightability test varied widely. Originality, in a few countries such as Germany, should be the result of high intellectual creation. [271] But in other countries, like the UK, the requirement of originality is not as high (as in Germany for example). Of course, this variation is not unique to computer software. It is true for copyright protection in

[266] Cheryl Cheung, "A Leading Canadian IP Case: Copyright for Computer Software" Deeth Williams Wall (13 March 2013), online: Deeth Williams Wall < http://www.dww.com/articles/a-leading-canadian-ip-case-copyright-for-computer-software>; see also J. Fraser Mann, "*Comment* on Apple Computer v. Mackintosh Computers*"* online: (1987) 32 McGill Law Journal 2 at 437ff < http://lawjournal.mcgill.ca/en/issue/1588>.
[267] *Supra* note 73.
[268] Sunny Handa, "Reverse Engineering Computer Programs under Canadian Copyright Law" (1994) 40 McGill LJ 621 at 627.
[269] George E. Fisk & Jane E. Clark, "Hardware and Software Protection in Canada" online: (1990) X 10 Computer L.J. at 483 http://repository.jmls.edu/cgi/viewcontent.cgi?article=1424&context=jitpl .
[270] Mindy J. Weichselbaum*, "*The EEC Directive on the Legal Protection of Computer Programs and U.S. Copyright Law: Should Copyright Law Permit Reverse Engineering of Computer Programs?" (1997) 3 Buffalo Journal of International Law 519 at 521.
[271] For instance, in German, the Federal Supreme Court said software is not copyrightable unless it is the result of personal creation, at Hoeren, in H.D.J. Jongen & A.P. Meijboom (eds.), *Copyright Software Protection in the EC* (Deventer/Boston: Kluwer, 1993), pp. 73ff; this point is of less relevant as, at least in EU case, there is directive which harmonizes those variations across member states.

general. Thirteen years after CONTU recommended copyright protection of software to Congress in the U.S, the EU Council adopted the directive on the legal protection of computer programs.[272] The directive was the result of three-year deliberation of the three EU highest bodies: the Commission, Parliament, and Council of Ministers.[273]

Harmonization and standardization of rules pertaining to computer programs across Europe is the main reason for this directive. Professor Samuelsson, recognizing the harmonization role of the directive, claims there is another secondary purpose for this directive: the need to bring EU software law in line with the United States law.[274] As far as harmonization in Europe is concerned, we have the following purpose clause in the directive:

Certain differences in the legal protection of computer programs offered by the laws of the Member States have direct and negative effects on the functioning of the internal market as regards computer programs.[275]

Existing differences having such effects need to be removed and new ones prevented from arising, while differences not adversely affecting the functioning of the internal market to a substantial degree need not be removed or prevented from arising.[276]

Akin to other specific EU rules, the directive on the legal protection of computer program is presented in a fairly detailed manner. It generally contains 11 articles and equally lengthy purpose clauses (preamble). To broadly highlight what has been included, it begins by clarifying the object of protection. The directive attempted, in Article 1, to delimit the proper scope of copyright protection, and makes a referral to the Berne Convention for the Protection of Literary and Artistic Works. Besides regarding computer programs as literary works, as addressed in Chapter One above, it stretches the reach of software protection to preparatory design materials. The EU copyright directive also has an exclusionary section. Article 1(2) excludes ideas and principles which underlie any element of a computer program, including those which underlie its

[272] *Supra* n.12.
[273] See Proposal for a Council Directive on the Legal Protection of Computer Programs, O.J. C 91/4 **(**1989); Amended Proposal for a Council Directive on the Legal Protection of Computer Programs, O.J. C 320/12 (Oct. 1990); and Council of Ministers, Common Position Paper, Art. 1, At 7 (Dec. 14, 1990).
[274] Supra note 71 at 279.
[275] *Supra* note 12, recital 4.
[276] *Ibid*, recital 5.

interfaces. Recital 11 of the preamble backs up this exclusion. This exclusionary Article of the directive seems to borrow the language used in 17 USC 102b.[277] Nonetheless, it does not define what it means. So, on its face, it leaves unresolved the issue as to whether "look and feel" such as screen shots, icons, menus, commands, and like objects that make up the user interface get protection. Protection only applies to expressions of ideas. This reflects the fundamental principles of copyright law that protect expressions of ideas, not 'ideas' themselves.

The directive in Article 2 and 3 defines right holders in terms of authorship. Part of the reason for the usage of this terminology is computer program in this directive is regarded as a literary work. Accordingly, the author of the program could be a natural person or legal entities. Article 2 (2 &3) has also recognized joint authorship and entitlement to economic rights by employers. Three important qualifications have been set out to bestow exclusive economic rights to employers. Firstly, employees should write programs in the execution of their duties. The other alternative is when they develop following the instruction of their employers. Finally, the entitlement of such exclusive economic right goes to employers if there is no contrary contractual agreement.

Notice should be made that the languages of computer programs are not protected. Programming languages are languages used to give instructions to computers.[278] In this regard, we have a leading case in Europe. This case is between *SAS Institute Inc. v World Programming Ltd*[279]. In the High Court of Justice of England and Wales, Mr. Justice Arnold rejected copyright claims on programming language and functionality of programs. However, he referred the case to the Court of Justice for the European Union (CJEU) for further clarification on the matter. The latter court basically found that programming language is not protectable. The court explained the issue in the following manner:

"Article 1(2) of Council Directive 91/250/EEC of 14 May 1991 on the legal protection of computer programs must be interpreted as meaning that neither the functionality of a computer program nor the programming language and the format of data files used in a computer program in order to exploit certain of its functions constitute a form of expression of that

[277] Greg Aharonian, "Deconstructing Software Copyright, 30 Years of Bad Logic", (2001) online: Internet Patent News Service < http://www.patenting-art.com/copyprob/softcopy.htm>.
[278] *Supra* note 34at 386.
[279] *SAS Institute Inc. v World Programming Ltd* , Case C-406/10.

program and, as such, are not protected by copyright in computer programs for the purposes of that directive.". "[280]

Experts praised the high court judge, the opinion of the advocate general and the highest court of EU position in excluding programming language and functionality (behavior) of programs from the reach of the directive.[281]

There is also early case law which clearly says copyright cannot protect the functionality of computer programs.[282] In *Navitaire Inc. v EasyJet Airline Co Ltd*[283] one English judge said:

"Copyright protection for computer software is a given, but I do not feel that the courts should be astute to extend that protection into a region where only the functional effects of a program are in issue. There is a respectable case for saying that copyright is not, in general, concerned with functional effects, and there is some advantage in a bright line rule protecting only the claimant's embodiment of the function in software and not some superset of that software".

Additionally, the directive in Article 4 laid out in a fairly detailed manner the specific rights copyright holders have. Generally, it grants three basic exclusive economic rights. Firstly, the author has reproduction rights – permanent or temporary reproduction, including loading, displaying, transmission or storage right. Adaptation, translation, and arrangement or other alteration of programs and reproduction of the result constitutes another exclusive right. The third exclusive right relates to distribution of the program to the public. However, the principle of exhaustion remains the limit for distribution right of authors of computer programs. Hence, the entitlement to control public distribution of programs benefits the author up until the point of first sale. What exactly is first sale in a digital context remains unclear. We have *UsedSoft GmbH v Oracle International Corp*, a very controversial decision handed down by the Court of

---

[280] *Ibid*. paragraph 71.

[281] Pamela Samuelson, Thomas Vinje, & William Cornish, "Does Copyright Protection under the EU Software Directive Extend to Computer Program Behavior, Languages and Interfaces?" online: (2012) European Intellectual Property Review <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1974890>; Jeremy Phillips, "Save Analytical Software"? That's not what SAS stands for…" The IPKat blog (January 2013) The IPKat blog, online: The IPKat: intellectual property news and fun for everyone < http://ipkitten.blogspot.ca/2013/01/save-analytical-software-thats-not-what.html>.

[282] *Navitaire Inc v EasyJet Airline Co Ltd [2004] EWHC 1725 (Ch.)* at para 178-185, *and Nova Productions Ltd v Mazooma Games Ltd [2007] EWCA Civ 219.*

[283] *Ibid*. para 94

Justice for the EU.[284] Oracle develops and markets software, mostly by offering programs online in a downloadable format. It does this with the use of license agreements, the most important terms of which include providing non-exclusive and *non-transferable* use rights for an *unlimited period* upon the payment of a one-off fee. On the other hand, UsedSoft offered for sale Oracle's *second-hand* software. Oracle lodged a lawsuit against UsedSoft in Germany. The Munich regional court decided in Oracle's favor, and UsedSoft appealed to the Federal High Court. The appellate court framed issues and referred the matter to Court of Justice for the European Union. The latter court, by disregarding Oracle's license agreement against exhaustion, said "use right for unlimited period is a sale" for the purpose of exhaustion.

Given the broad rights granted to right holders under Article 4, it is imperative to have a clear definition of what users may legitimately do with the programs. With this purpose in mind, the directive in Article 5 introduces the notion of lawful acquirer and what he can and cannot do. Accordingly, users are entitled to do three important acts without authorization of the right holders. These are:

- ✓ The making of a back-up copy by a person having a right to use the computer program may not be prevented by contract in so far as it is *necessary* for that use. In this regard, determining what is necessary and what is could be a painstaking task, and main source of litigation.
- ✓ 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. This act is short of decompilation.
- ✓ Doing acts specified under Article 4 for the purpose of error correction.[285] However, these user's or lawful acquirer's rights will kick in as much as they are necessary for a particular purpose.

The directive also introduced the concept of decompiling for the sake of creating interoperability

[284] *UsedSoft GmbH v Oracle International Corp* , Case C-128/11.
[285]Supra note 12, article 5(1, 2 & 3); However, sub (1) has another proviso which reads as…in the absence of specific contractual provisions. It means this exception may be overridden by contract. This will again raise another interesting tension with recital 16 of the preamble –which can be read differently.

of an independently created computer program with other programs.[286] Sub-Article 3 of Article 5 seems to allow very limited forms of reverse engineering. If we view the sub-article as a whole, it is very difficult to get a firm grip on it. But the argument that this sub-article is about reverse engineering may raise a problem of a carefully worded concept of decompilation under Article 6. What article 6 does is allow decompilation only for the purpose of achieving interoperability; *i.e.*, interoperability with some other programs. The rights to reverse-engineer set out in Article 6 are very restricted.[287] One such restriction is the requirement of necessity. Decompilation is also not allowed if the information necessary to achieve interoperability has previously been readily available.

2.3 Requirements for software copyright protection

The other most important aspect, in discussing the legal protection of computer software is the requirements for its protection. This is particularly relevant in the copyright area. Though not complicated like patents, there are substantive and procedural requisites of copyright protection. The substantive requirement concerns the broad and open-ended appropriate subject matters warranting copyright protection.[288] Computer programs being literary works, meet the subject matter requirement. What come next are the requirements of originality and fixation. All copyrightable works have to be original in the sense that they should result from the effort of the author – creating a nexus between the work and the author (not copied from somewhere else). Originality concerns expression of ideas [programs] not the ideas [function of the software].Works need not be of a "never before" kind.[289]

However, some kind of intellectual involvement is required. In Canada, originality is not defined in the *Act* although section 5 seems to provide some clues. In *CCH Canadian Ltd. v. Law Society of Upper Canada*,[290] the Supreme Court of Canada clarifies the extent of originality for copyrightable works. McLachlin CJC, writing the decision, said the following:

---

[286] *Ibid*, article 6
[287] This restriction is also indicated in recital 15 of the preamble
[288] See, for instance, 17 U.S.C § 102; *supra* note 73 s. 5.1
[289] See, especially Carys J Craig, "The Evolution of Originality in Canadian Copyright Law: Authorship, Reward and the Public Interest", (2005) Osgoode Hall LJ 425 at 429.
[290]Supra note

For a work to be "original" within the meaning of the Copyright Act, it must be more than a mere copy of another work. At the same time, it need not be creative, in the sense of being novel or unique. What is required to attract copyright protection in the expression of an idea is an exercise of skill and judgment. By skill, I mean the use of one's knowledge, developed aptitude or practiced ability in producing the work. By judgment, I mean the use of one's capacity for discernment or ability to form an opinion or evaluation by comparing different possible options in producing the work. This exercise of skill and judgment will necessarily involve intellectual effort.[291]

Hence, to pass this test, computer programs should result from the intellectual effort of developers. In other words, the involvement of some sort of skill and judgment is necessary. The EU software directive seems to have a similar clause. Section 1(3) of the directive requires programs to be the intellectual creation of authors. The directive goes further and says no other criteria are applied to determine the copyright protection of computer programs. In the U.S. too, section 102(a) spells out the above two requirements. As stated, the general rules of copyright apply to computer programs in Canada and the U.S. We rarely find court cases particularly addressing originality and computer software. Accordingly, it is important to examine relevant jurisprudence on other literary works so that we can apply the standard to computer programs. The U.S Supreme Court's decision in *Feist Publ'ns, Inc. v. Rural Tel. Serv. C*[292] is particularly important as far as the originality requirement is concerned. It is about selection, organization, and arrangement of data otherwise referred to as compilation. Rural Tel. organized its customer lists in alphabetical order, which is ordered by law. Feist Publishing Corp. took raw facts (telephone directory) from the Rural Tel. The latter brought a copyright infringement claim. The Supreme Court said information in a rural directory is not copyrightable as one cannot find independent creation of the work on the part of the telephone company. *Feist* sets out many copyright principles, and the part relating to the analysis of the constitutional clause is fascinating. The court stated that the purpose of copyright is not rewarding mere efforts. It rather intends to encourage creative expression.

[291] Ibid at par. 24
[292] Feist Pubs., Inc. v. Rural Tel. Serv. Co., Inc., 499 U.S. 340 (1991)

The other requirement for copyrighting a computer program is fixation. Generally speaking, fixation is not a statutory requirement. Even the Berne Convention gives discretion to member states.[293] The EU software copyright laws do not incorporate fixation requirement. However, the definitional section of the Copyright *Act* of Canada spells out fixation requirement of copyrightability by stating a computer program must be "expressed, fixed, and embodied or stored in any manner."[294] One notable Canadian court case, denied a copyright infringement claim based on insufficiency in fixation criteria. The case is *Canadian Admiral Corporation Ltd. v. Rediffusion Inc.*[295] On the other hand, the U.S. law is very clear about fixation requirement is concerned. Section 102 (a) of the Copyright *Act* reads "...fixed in any tangible medium of expression…"[296]

2.4 Trade secret protection of computer software

Trade secret protection was important during the mainframe and minicomputer eras – as software was rarely distributed in its source code form. It is and was an ideal mechanism to protect the internal working and design of software source code. That remained true until the beginning of the PC era, which means that around the year 1990 and over the next couple of decades, the importance of Trade Secret declined somewhat. This decline could be attributed to two reasons.[297] Firstly, copyright protection rose to the forefront as a dominant paradigm. The scope of copyright in relation to computer software broadened in this period. There was an aggressive attempt to protect the structural aspects of software that previously would have been protected by trade secret law. Secondly, with the rise of WWW in the 1990s, much of the functional coding behind web pages was generally made visible.

That decline began to level out, however, for the launch of Salesforce, a cloud computing company, in 1999, and we moved into cloud computing. For decades, we can say the use of trade secret protection in the software industry remained at a constant and important level. Hence, trade secret protection for the hidden part of software has not been overly controversial. It has

[293] *Supra* note 10. Article 2 (2) reads*: It shall, however, be a matter for legislation in the countries of the Union to prescribe that works in general or any specified categories of works shall not be protected unless they have been fixed in some material form.*.
[294] *Supra* note 73.
[295] *Canadian Admiral Corporation Ltd. v. Rediffusion Inc*, [1954] Ex. CR 382, 20 CPR 75
[296] *Supra* note 72.
[297] *Supra* note 86.

always protected source code, the interworking of software – things we cannot see. These days, trade secret specialists propose expanding its scope – arguing for the possibility of protecting the *revealed* aspects of software.[298]

[298] See generally, Michael Risch , "Hidden in Plain Sight" , Online: (2o16) Villanova Public Law and Legal Theory Working Paper Series, < https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2761100##> .

CHAPTER THREE

FLAWS OF THE EXISTING SYSTEM AND SOME BALANCING EFFORTS

3.1 Introduction

The increasing expansion of computing and software technologies is a reality. Transactions are becoming increasingly virtual. Software technologies play a greater role in those transactions. This chapter contains four parts. In part I, I argue the industry sensitive nature of existing software protection laws. Part II discusses strong criticisms forwarded to three of the intellectual property protections of computer software. More particularly, this part analyses the inapplicability of copyright, patent and trade secret laws to computer software. Section III covers some balancing attempts in the current system. Accordingly, I discuss the open and free software movements and their implication in ensuring the interest of the public. Additionally, the limited instances of reverse engineering of software and the extent of its permission in the EU and U.S. form another balancing attempt. Finally, the paper argues that software is unique, and discusses the distinctive nature of computer software.

3.2 The "Cherry Picking" Nature of Current Intellectual Property Laws and Practices

The preceding two chapters testify to numerous facts about the existing protection of computer software. Firstly, they show the overprotection aspect. We see multiple protection mechanisms which amount to overprotection.[299] Computer programs enjoy almost all traditional forms of intellectual property rights. It is popularly believed that copyright forms the conventional software protection mechanism. Additionally, thousands of patents are/being granted by patent offices. This is particularly the case in the U.S., which is the leading nation in the software industry. Significant software patents have been granted at the European level, too. Undoubtedly,

[299] Mark M. Friedman, "Copyrighting Machine Language Computer Software-The Case Against", online: (1989) 9 Computer L.J. 1 at 2 http://repository.jmls.edu/cgi/viewcontent.cgi?article=1430&context=jitpl ; Some even think one form of intellectual property could over protect software let alone multiple protections. For example, Samuelson and her team believe copyright is over protecting computer software (See, supra note 23, at 2359).

trade secret has served as one of the trustworthy computer software protection paradigms for decades. Software companies also use trademarks to protect some elements of computer programs. Furthermore, other private contractual (licensing) mechanisms and self- regulations (e.g. technological protection) are widely used means of protecting computer programs.

Some authors have gone further and argued for *sui generis* protection as the best or better way of protecting software.[300] Such a mode of protecting software was not a novel recommendation, as the *WIPO* has suggested a similar recommendation in the1970s.[301] The international bureau of WIPO has adopted Model provisions for the protection of computer software, the main goal being to assist in creating certainty.[302] Needless to say, computer software requires strong protection as it is quite vulnerable to piracy.[303] Nonetheless, stricter and effective protection does not mean overprotection. Applying patent, copyright, trade secret and other laws to protect software equals overprotection. Currently, software is getting all those forms of protection.

Generally, software companies have been demanding every possible protection mechanism for the last six decades. They do that through the help of lawyers and Software Company funded researchers. Of course, it is true that software is unique and important. There is no dispute of this proposition. Yet, although software is unique and invaluable, this does not mean it should enjoy the protection of all intellectual property rights. These days, it is hardly possible to find an intellectual property law that does not protect software algorithm. This is not the case in the other protectable subject matters. Some authors even believe that intellectual property rights are mutually exclusive by their nature.[304] Let us see the mutually exclusive nature of the three of intellectual property rights. Patent protects ideas while copyright protects their expressions. Trade secret seeks the confidentiality of commercial information. In contrast, patent is known for

---

[300] *Supra* note 11 at 187; for general understanding of this proposal, see John C. Phillips, supra note 25; & Steven B. Toeniskoetter, "Protection of Software Intellectual Property in Europe: An Alternative Sui Generis Approach", online: (2007) 10 INTELL. PROP. L. BULL 65 at 76 <http://heinonline.org/HOL/Page?handle=hein.journals/iprop10&div=9&g_sent=1&collection=journals> [He paralleled software and database protection and proposed a European wide sui generis software protection].

[301] *Supra* note 29, (Pamela Samuelson and her colleagues) at 2312. It was a 13 articles length draft model law for software. This model provision defined software, and it proposed a twenty years protection.

[302] Trevor Cook, ed., *Sterling on World Copyright Law*, 4th ed (London, UK: Sweet & Maxwell, 2015) at 1679

[303] Mickey T. Mihm, "Software Piracy and the Personal Computer: Is the1980 Software Copyright Act Effective?", (1983) 4 Computer L.J. 1 at 171-193; supra note 6.

[304] *Supra* not 215 at 16. However, there are overlapping scenarios in the intellectual property world although very different in many respects from the software case. For further appreciation of this matter, see Neil Wilkof & Shamnad Basheer, eds, *Overlapping Intellectual Property Rights*, (Oxford, U.K :Oxford University Press, 2012)

its disclosure doctrine. Copyright does not bother with the utility of works, and their functionality is not an issue. Instead, a patent protects functionality and weighs the utility aspect of patentable subject matters. This implies those intellectual property rights are unique and protect different aspects of creations or inventions. The question then becomes: why are computer programs enjoying the protection of patents, copyrights, trade secrets, trademark and trade dress laws, design laws, *etc.?*

Theoretically speaking, the utilitarian justification seems the main basis of computer software protection. At the heart of this utilitarian justification is the promotion of the public interest. It is thought that an intellectual object does something important for the wellbeing of society. The best example in this regard is the United States IPRs system. Unlike many, if not all, jurisdictions of the world, the U.S. Constitution has a clause about the justification of intellectual property. It authorizes Congress *''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''*.[305]

A closer examination of the software directive of EU also reveals the implicit recognition of the utilitarian approach the European Union has adopted. It says that protection of computer program in Europe is also given for "*industrial development purpose"*.[306] Member states of the EU also had this theory incorporated in early IP statutes. For instance, the early copyright law of U.K, statutes of Anne[307], shows us that incentive theory has been recognized since the 18th century– encouragement of learning.

The other assumption is that creation requires an investment of labor, as pro-labor theory writers claimed. In addition to this, creation and invention require an investment of time, money, and training or education. More specifically, the invention requires a huge investment of money. People will not make an investment of effort, time and money unless there is a legal regime which gives an opportunity to pay off these investments. There will be an incentive to produce goods because their selling price will allow a producer [creator or inventor] to recoup both the costs of production and the benefit of the goods to a purchaser.[308] For economic theorists, the intended

[305] Article I, paragraph 8, cl 8.
[306] Supra note 12, recital 3.
[307] *Statute of Anne of 1709* (U.K), 8 Anne, c.21.
[308] *Supra* note 111 (Tanya A. & Jennifer D) at 51.

beneficiary of [software product or service] is the community as a whole, which demands the production of, and *access to*, as many creative works as possible.[309]

The main goal of the U.S.'s constitutional clause, based on the Supreme Court's interpretation in *Twentieth Century Music Corp. v. Aiken*[310], is *"by this incentive, to stimulate artistic creativity for the general public good."*[311] The public interest role of intellectual property rights has been pointed out even in much older cases. For instance, in *Washingtonian Pub. Co. v. Pearson*, the court stated that the ultimate aim of granting patents and copyright was to provide lasting benefit to the world.[312] The U.S. Supreme Court in *Mazer v. Stein*,[313] also said:

"The economic philosophy behind the clause empowering Congress to grant patents and copyrights is the conviction that encouragement of individual effort by personal gain is the best way to advance public welfare through the talents of authors and inventors in "Science and the Useful Arts." Sacrificial days devoted to such creative activities deserve rewards commensurate with the services rendered."[314]

We have to carefully examine this assertion. For one thing, the law wants to encourage individual effort, and one can say this is an application of labor theory – so that the right holder [programmer] uses all sort of rights to prevent access. This is not an entirely incorrect assumption. But encouragement of individual effort by personal gain is intended "to promote the progress of Science and useful Arts", or in the words of the Supreme Court, '*to advance public welfare'*. By personal gain, the court means intellectual property rights or similar protections. The grant of the right to creators or inventors is not an end. It is rather an incentivizing means, spurring creativity and innovation. The court in *Feist Publication, Inc.* supports this notion, as it says *"the primary objective of copyright is not to reward the labor of authors, but to promote the Progress of Science and useful Arts."*

---

[309] *Ibid*.
[310] *Twentieth Century Music Corp. v. Aiken*,422 U.S. 45 L. Ed. 2d 84, 95 S. Ct. 2040 (1975);see, also *United States v. Paramount Pictures, Inc*., 334 U.S 131 (1948) [Here, it seems primacy is given for advancement of science, innovation and human knowledge than recognizing proprietors].
[311] *Supra* note 315 *at 151, 156.*
[312] *Interstate Circuit, Inc. v. United States ,306 U.S.* (1939) at par. 30; *Mazer v. Stein, 347 U.S.(1954)*, at par. 219
[313] Ibid, *Mazer v. Stein, 347 U.S.(1954)*.
[314]*Ibid,* at Par 201..

This purpose may have many facets. Firstly, it could mean the subject matters created or invented should play a utilitarian role. This facet concerns the aesthetic function of works or the utility of inventions. Patent seems to meet the utility limb, as patent utility is one statutory requirement for the grant of patents. On the other hand, usefulness is not a prerequisite to copyrightability. The second limb concerns the *access issue*. The substantive usefulness of the protected work or invention is by no means sufficient. Only when the public utilized it can we say it is for the public good. That means if it is *inaccessible* for many reasons, the main purpose of protection misses its point. In the case of software, there is no question of its utility in this networked era and information economy. However, we see that the overprotection of software seriously undermines the *public good* purpose of intellectual property laws. Intellectual property rights are restraints on competitors, and affect consumers' wide access need. Software IP even results in much more impact to the consumer as software innovation affects almost all aspects of consumer's life.

In most cases, the effort to ensure the interest of the public/consumers while safeguarding the interests of intellectual property owners/holders is a daunting task. In other words, these two interests are at odds. The right holder wants broader, lengthy and stronger protection. In contrast, consumers demand *access* – wider dissemination, less expensive, shorter protection, broad exception, and open access. In software cases, broader and overprotection on the one hand, and over-emphasis on the incentive role of intellectual property rights to computer software on the other, seriously impacts the interests of consumers of digital products [software users]. Peter S. Menell noted this phenomenon and described it:

The peculiar nature of the public goods problem with regard to computer software and the network externality inherent in computer systems, however, breaks the neat link, in the typical case, between broad protection and the inducement of the optimal level of innovation to promote the public interest.[315]

Professor Menell in criticizing CONTU's analysis identified three major problems since CONTU recommended copyright to a computer program.[316] The first and foremost problem is the barriers to small entrants. Secondly, developing non-infringing and compatible programs costs vast

[315] Peter S. Menell , Tailoring Legal Protection For Computer Software*"*, (1987), 39 Stan L Rev 6 at 1330.
[316] *Ibid.*

resources. Lack of clarity of standards in the software industry is the third problem. He believes these major problems can be attributed to the report's failure to appreciate two significant aspects of computer software: the problems to the public good that software poses, and its unique characteristics.[317]

The United States Congress created CONTU to further study and recommend legal protections of digital works, including computer software. In doing so, Congress underlined that the recommended mechanism should ensure public access to those digital technologies.[318] Accordingly, CONTU recommended copyright as a protecting mechanism for computer software. Although many challenge the copyrightability of computer software, as has been mentioned above it is not only copyright that protects software. The laws seems only be adopted to benefit software and technology companies.

If we regard software as copyrightable subject matter, then we are considering it as a literary work. Once the developer satisfies the requirement of originality of expression and in some jurisdictions, fixation, then protection is given for longer periods of time than are typical for patents.[319] The rights holder, therefore, will enjoy an exclusive right to produce, reproduce and distribute the program for the protection period. Someone else with permission could write his or her program without violating the protection afforded by the copyright.[320] But, unlike patent protection, the danger here is that other developers can also come up with the same program (functionally) independently without permission of the previous developer.

If we apply the regime of patent protection to software, there is a situation wherein it benefits both the developer and the public at large. The patentee will have the monopoly on a patented software invention[321] and enjoy an exclusive right. At the same time, the public at large will find out the owner and the scope of the right[322] as the grant of a patent requires disclosure and specification of rights in the form of a "claim(s)". Besides, the disclosure of the way the

[317] Ibid
[318] Ibid at 1329
[319] In Canada it lasts, for individual author, for 50 years and in US and EU life of a person plus 70years.
[320] Bernard A. Galler, *Software and Intellectual protection: Copyright and patent Issues for Computer and legal Professionals*, (London, UK: Quorum Books, 1996) at 12.
[321] Henry Carr and Richard Arnold, *Computer Software: Legal Protection in, 2nd ed.,(London, UK* : Sweet & Maxwell, 1992) at 127.
[322] *Ibid.*

software is developed will enable the public to redevelop after the lapse of patent protection.[323] So, the rights in those specific programs will be mentioned in the claims of the patent application. Such an approach resolves the litigation on reverse engineering and decompilation that will be discussed below.

3.3 Abandoning the Current Legal Framework

The existing legal framework regarding computer software is full of uncertainties. Completely disregarding the current laws and approaching software regulation afresh may be a painstaking task (in terms of resource, time etc.), but as one cannot make an omelet without breaking eggs, we have to reconsider the existing system. Particular consideration should be made of the effect of software on societal life and today's reality.

There are many stakeholders in the software world (its development, uses and consumption). We can at least discern three interests: the interest of the incumbents (huge hardware and software companies); small entrants and individual software developers; and consumers of computer software. Making software amenable to all forms of traditional intellectual property laws only considers one of the aforementioned interests. That makes the system problematic, at least as seen from the very purpose of IPRs and the other two interests' point of view. It is not only the multiple levels of protection that troubles the most. The extent and scope of each form of protection are the subject of ongoing contentions. As regards the interest of new entrants, gigantic hardware and software companies own thousands of patents and use other forms of IPRs, and impact the competition.

The existing system does not properly address, among other things, the following issues. The first relates to the availability and affordability issue. Developers of computer programmers do not produce software out of anything. In the age of the internet, everyone is online, and all share information. Hence, the cost they are speaking of and the profit they are making do not seem reasonable, or require further study. They should only get what they deserve. This takes us to the theoretical debates of intellectual property rights (the need to identify right holders' investment and what they gained from the common pool). Secondly, the compatibility and interoperability

[323] After the expiry of the patent protection the invention will form public domain. In some jurisdiction patent protection lasts for 20 years and in others expires shorter.

issue is not less important. There are multiple categories of electronic machines that consumers use in the market (from a smart watch to mainframes). Hence, it is important to have a rule addressing compatibility and interoperability of software to variants of devices. It seems we have limited rules and case law on this point, but they are not adequate and, at times, not clear. Thirdly, there is the question of adaptability: computer software is adaptable by its nature.[324] These natures coupled with its unique features discussed below necessitate the reconsideration of the existing system.

3.3.1 Copyright misfits computer programs

The copyrightability of computer software seems a fairly settled matter. However, the different tests courts have developed and the approaches countries took prove the conventional understanding wrong. There are still many challenges as regards the copyrightability of computer software and its infringement issues.

One bold contention frequently raised is that of its copyrightability itself. Professor Samuelson and her team believe that copyright is not suitable to a computer program. They regard a computer program "*as a virtual machine and as a medium of creation*"[325], and reject the assumption that software codes constitute "*literary texts",* hence non-copyrightable. Greg Aharonian has also argued for the abolition of software copyright in light of *17 USC 102b* and its equivalents.[326] The first reason for proposing this is *"copyright is a bad law with no logical basis in the mathematics and physics of information processing*."[327]

Moreover, copyright largely protects arts: writings and creative works of aesthetic value. On the one hand, it achieves the utilitarian purpose – dissemination of information to the public and recognition of the interest of the creator of the work. The recognition may relate to the expression of personhood (as in the case of novels, musical or dramatic works) or economic interest, or some other moral interest. Despite that, there is a limited element which constitutes the literal element, as computer software is a technological output which falls into the category

[324] Pamela Samuelson, "Modifying Copyrighted Software: Adjusting Copyright Doctrine to Accommodate a Technology", online: (1988) 28 Jurimetrics J 179 < http://scholarship.law.berkeley.edu/facpubs/653/>
[325] Robert **A.** Gorman, Comments on a manifesto concerning on the protection of Computer Programs, (1994-1996) 5 Alb. L.J. Sci. & Tech. 277 at 279
[326] *Supra* note 282.
[327] *Ibid.*

of basic science. It is the work of engineers [programmers], and the question of whether it forms an art or not has remained an ongoing contentious issue.[328]

The CONTU report magnified the piracy problem and proposed copyright to guarantee the protection of the huge investments made by hardware and software companies. Copyright may serve this purpose by preventing the direct copying and usage of similar lines of codes. However, it does not prevent among others, *"the use of incorporated algorithms, ideas, and designs".*[329] *The software directive in EU, too, excludes "Ideas and principles which underlie any element of a computer program, including those which underlie its interfaces."*[330]

Computer programs involve a great amount of mathematics.[331] Certain patent claims have been rejected based on mathematical formula criterion. What about copyright? Mathematical concepts are not copyrightable under TRIPS agreement.[332] On the other hand, the agreement explicitly allows copyrighting of program source and object codes of any form.[333] What if these codes engage mathematics to a greater extent?

In copyright law, there is such a thing called derivative works.[334] For instance, translations, adaptations arrangements, modifications and other transformations of works are regarded as a derivative work, and copyright subsists for these works. What if a programmer converts one computer language or code into another? Is it like translating a novel or poem from one language to another? Is there such a thing called derivative works in computer software? [335] Further

---

[328] Information Society Technologies Advisory Group, "The Missing Key Enabling Technology Toward a Strategic Agenda for Software Technologies in Europe" , online: (2012) EU Commission at 10 < https://ec.europa.eu/digital-single-market/en/news/software-technologies-missing-key-enabling-technologies-istag-working-group-software>.
[329] Hannes Westermann, How to treat software in the intellectual property framework (LLM thesis, Lund University Faculty of Law, 2016) [unpublished] at 7.
[330] Supra note 12, article 1(2).
[331] *Supra* note 30 at 147.
[332] *Supra* note 8, article 9(2).
[333] *Ibid*, article 10.
[334] *The Berne Convention, Art. 2 § 3(though it does not particularly use the phrase derivative works, it stipulates that "Translations, adaptations, arrangements of music and other alterations of a literary or artistic work shall be protected as original works without prejudice to the copyright of the original work.");The TRIPS agreement in article 9(1) has also incorporated Art 2, § 3 of Berne; U.S., 17 U.S.C § 101(defined derivative works) , 103(b) & § 106(2);* For Canadian perspective, See generally, William J. Braithwaite, "Derivative Works in Canadian Copyright Law", (1982) 20: 2 Osgoode Hall LJ 192; However, some authors have criticism to such categories of works. See, for instance, Daniel Gervais, "The Derivative Right, or Why Copyright Law Protects Foxes Better than Hedgehogs", (2013) 15: 4 Vand J Ent L & Prac 785
[335] See US case *Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 609 F. Supp. 1307, 225 U.S.P.Q. (BNA) 156 (E.D. Pa. 1985*. The court said-'….transferring or converting from one computer language to another is not comparable to translating a book written in English to French'.

discussion is needed on the application of certain derivative works in the doctrine of reverse engineering.

3.3.2 Patent is inapplicable to computer software

With reference to patents, there is no explicit legislation which allows for software patentability. Though unsuccessful, there was an attempt to adopt a patent law to computer-implemented inventions (software inventions) in Europe.[336] Most patent laws even exclude software from the subject matter of patent. For instance, the European patent convention excludes patenting computer program *as such*.[337] Courts and patent offices interpreted this exclusion, and introduced many criteria and tests, including the technical nature and contribution test. However much neither the legislation explicitly allowed patentability nor courts clearly pronounced their patentability, many software patents have been issued.

Software is all about mathematics and logical sequencing of algorithms. It also involves a lot of abstractions. This renders software unpatentable. Some believe software innovation requires little investment, so patents are not needed to promote this type of innovation.[338] The other factor that makes software patents unfeasible is their term. In most cases, the span of patent term runs for only 20 years. On the other hand, patent prosecution and application take much time. The short-lived nature of software makes patents unsuitable to software. Some of them last for only weeks and months.[339]

Copyrighting software generally believed to be the accepted rule. We normally copyright expressions of ideas, not ideas themselves. The other cousins of copyright, patents protect ideas rather than expressions of ideas. The question then is: why are we patenting expression of ideas, if patent is meant to protect inventive ideas rather than expressions? Professor Kospsell questions the categorization into patents and copyrights at all, and alternatively suggests that software is a "hybrid" object.[340]

---

[336] *Supra* note 20.
[337] *Supra* note 201, article 52 (2(C)) and (3).
[338] Robert E. Thomas, "Debugging Software Patents: Increasing Innovation and Reducing Uncertainty in the Judicial Reform of Software Patent Law", (2008) 25 Santa Clara Computer & High Tech. L.J**. at** 193.
[339] *Supra* note 86**;** *supra* note 187 at 116.
[340] *Supra* note 215 at 33.

3.3.3 Challenges of trade secret protection of computer software

Software protection through trade secret law may have been considered feasible in the past. Then, technology was not widely distributed, and the issue of trust is not an issue. Conversely, today's reality is different. We see the wider distribution of computer and software, in the words of Laurence Diver[341] –"everywhere from the African shanty town to trading floors of Wall Street". The more software is distributed, the greater the possibility of disclosing confidential information. In addition, we see the growing movement of labor in today's integrated economy, and there is no international trade secret treaty, unlike other forms of intellectual property rights.[342]

Trade secret law is not in accordance with the open source and free software movements. The latter group advocated for the wider availability and accessibility of software codes. The practice of reverse engineering and decompilation also do not go with the notion of trade secrets.

3.4 The Special Nature of Computer Software

Software has many unique characteristics. Though software is copyrightable subject matter, its literary nature is questionable. Unlike other literary works, software affects every aspect of today's world. It goes without saying that, its complex nature and technological aspect makes software unique from other copyrightable works and patentable subject matters. The following sub-sections discuss the unique features of software.

3.4.1 Software is not merely a literary work

Computer software is more than a traditional literary work. It is a technology[343], too. It is a technology that touches every aspect of human life. Software is becoming increasingly indispensable in the information society era. Considering only copyright, software is unique. That means if we take copyright law and look at what makes software copyrightable, we find few attributes of software copyrightability. There is less confusion as to the distinction between software and other copyrightable subject matters than literary works. One rarely finds commonality among dramatic, musical and artistic works and software. Can one quote few lines of software instructions, and properly reference them? We know that through the fair dealing

[341] *Supra* note 34 at 125.
[342] *Supra* note 226 at 1423.
[343] *Supra* note 246 at 65.

exception we can use part of someone's work as long as we give the proper recognition in a proper format. That does not seem to work on computer software world, at least legally speaking. Does that mean computer programmers entirely create something new out of nowhere? The answer perhaps is "No".

3.4.2 Software is ubiquitous

Fifty or sixty years ago, there were only limited computers in the market. In recent years, however, the computing market increased dramatically. Every aspect of our life is tied in with software. Almost all sectors (public and private) use computing technologies in their everyday transactions. As a result of this, we find software everywhere. The omnipresent[344] nature of software makes it unique and warrants a special regulation. At the very least, the ubiquitous nature of software makes the application of traditional intellectual property laws ill-suited.

3.4.3 Complex nature of software

Computer software is a complicated concept. This complexity remains one of the leading factors for the existing uncertainty of its protection in patent, copyright or/and trade secret.[345] It is very difficult for an ordinary consumer to appreciate what amounts to software infringement and what does not. Some of its parts are very complex. We may understand the source code aspect of computer software. However, the object code aspect is not even intelligible to expert programmers. The combination of binary numbers made it only susceptible to machine understanding. Besides, software development involves different technologies of translation. The different programming languages, assembler, compiler, and translator are a manifestation of the complex nature of software.

The task of determining what does it include and does not raises an array of difficulties. Concepts such as source code, object code, structure and organization of source and object codes, micro codes, disk operating systems, programs running behind the screen, user support documents (textual document and training), look and feel (the way screens interact with each other), the organization and interaction of a program's function, and macro code, require specific regulation. The type of computer software, too, varies, depending on, among other things, its

[344] *Supra* note 34at 126.
[345] Howard K. Szabo, "International Protection of Computer Software: The Need for Sui Generis Legislation", (1986) 8 Loy L.A. Int'l & Comp. L. Rev511 at 515.

function, what companies produce and write them and who consumes them. For instance, we have packaged software[346], custom software application, and embedded software application.[347]The complex nature of software makes it different and unique from literary works, let alone from an extensive list of other copyrightable works.[348]

3.4.4 Codes regulating software codes

The concept of "the internet of things, " raised above, and the advent of new technologies and connected content[349] can cut both ways. Firstly, it improves the access rights[350] of consumers by easily facilitating the dissemination of information and commercialization of works. At the same time, it also creates problems for copyright holders[351] by enabling "*pirates to steal efficiently."*[352] Also, it is difficult to control[353] and trace technologies, if not impossible. Hence, the expansion of those technologies exacerbates the enforcement problem even more. As the development of digital media and computer technologies is creating difficulties in enforcing copyright laws, industry is devising self-enforcement mechanisms.[354] This is true for digital intellectual objects.

---

[346] Report of an Industry Expert Group on a European Software Strategy, Playing To Win In the New Software Market: Software 2.0: Winning For Europe, (June 2009 Version 3.5)

[347] Ibid; For further appreciation of complexities as to what a computer program is and how they work see, Pamela Samuelson, "CONTU Revisited: The Case against Copyright Protection for Computer Programs in Machine readable Form", (1984) *Duke LJ 663* at 672-681

[348] Berne Convention for the Protection of Literary and Artistic Works, *opened for signature* Sept. 9, 1886, 828 U.N.T.S. 221, **S.** Treaty Doc. No. 99-27, 99th Cong. (1986) (revised at Paris, July 24, 1979) article 2(1) enumerated extensive copyrightable works; Trevor Cook, ed., Sterling on World Copyright Law, 4th ed (London, UK: Sweet & Maxwell, 2015) at 262-284; Hector MacQueen et al, Contemporary Intellectual Property Law and Policy, 2nd ed (Oxford, UK: Oxford University Press,2010) at 45, 62-87 ; and *Lionel Bently & Brad Sherman, Intellectual Property Law*, 3rd ed (New York, U.S.A: Oxford University Press, 2009) at 59-90 [these books spell out the UK version of copyrightable works. It, among others, categorizes these works in to Literary works (novels, short stories, poetry, song lyrics, non-fiction books, periodical articles, computer programs, database, circuit diagrams), dramatic works (e.g. dances, mimes),musical works, artistic works(e.g. graphics. paintings, drawings, photographs, sculptures, collage, works of architectures, works of craftsmanship); To appreciate the Canadian context, *see* D. Jeffrey Brown & Marisia Campbell, "Copyright" in Stuart C. McCormack, ed, *Intellectual Property Law of Canada 2nd ed (New York, U.S.A: Juris Publishing, Inc., 2010) at 218-229; supra note 51 (*Sunny Handa) at 155-181

[349] See, Graham Reynolds, "*Towards a Right to Engage in the Fair Transformative Use of Copyright-Protected Expression", in Michael Geist, ed, From "Radical Extremism" to "Balanced Copyright": Canadian Copyright and the Digital Agenda* (Toronto: Irwin Books, 2010) at 395.

[350] Brad Sherman and Leanne Wiseman, ed, *Copyright and the Challenge of the New*, (The Netherlands: Kluwer Law International, 2012) at 7.

[351] *Ibid*.

[352] Robin Andrews, "Copyright Infringement and the Internet: An Economic Analysis of Crime", (2005)11:2 BUJ Sci. & Tech L.

[353] *Supra* note 23.

[354] For the better appreciation of what the digital environment poses to copyright enforcement, *see* Sandra V.I. Scmitz, *The Struggle in Online Copyright Enforcement: Problems and Prospects* (Luxemburg: Hart Publishing, 2015) at 30, 56.

The copy and print control Digital Right Managements Systems (DRMS), employed by High Tech Corporation, is an example in this regard. Although the existing systems allow for the multiple protection of software, there is also widespread piracy[355] of artistic and literary [software] works for which the authors of these works complain to the public and the government. They are going in the direction of an approach where, to use Charles Clark's expression, **"***the answer to the machine is in the machine"* "[356]. Lessig, in his book *Code and Other Laws of Cyberspace*[357], succinctly addressed the private regulation of digital copyrightable materials. This private regulation for him is referred to as *Code*. Reidenberg [358] referred to this regulatory mechanism as "*network architecture".* Modern copyright instruments give legal effect to these technological protection mechanisms (encryption, copy and access control mechanisms).[359]

3.4.5 The application of first-sale principle

Normally, the exclusive right to use IP right ends up on the first sale of that specific subject matter, be it a patentable product, process, or copyrightable work (artistic, literary, musical, and dramatic works). For instance, the copyright owner loses control over their copies of specific work upon getting the required remuneration from the user. Here, the most important fact is determining whether the work is put on the market with the consent of the right holder. Thereafter, the user can freely use or further transfer that specific work to other users. The original owner has no right to interfere, with the exception of moral rights related instances. This is the general rule for all copyrightable works. It is called the first sale doctrine in some countries such as the U.S., and exhaustion principle in others such as the EU. The exhaustion principle

[355] For a broader understanding of modern copyright statutes and the piracy problem, see Trajce Evetkovski, *Copyright and Popular media: Liberal Villains and Technological Change*,(London, UK: Palgrave Macmillan, 2013) at153-162); Sterling on World Copyright Law, at 7, 33.
[356] Charles Clark, "The Answer to the Machine is in the Machine", in P. Bernt Hugenholtz, ed, *The Future of Copyright in a Digital Environment*, (The Hague: Kluwer Law International, 1996), at 139.
[357] Lawrence Lessig, *Code: And Other Laws of Cyberspace*, (New York, U.S.A: Basic Books, 2006). Lessig has also reinforced the comparable regulatory role of *codes* in his other works, see, Lawrence Lessig, "Law Regulating Code Regulating Law*"*, (2003) 35 Loy. U. Chi. L. J. 1 1-14; and Lawrence Lessig, "The Law of the Horse: What Cyber law Might Teach", (1999) 113 *Harv L Rev* 501-546.
[358]Joel R. Reidenberg, "*Lex Informatica*: The Formulation of Information Policy Rules through Technology", online: (1998) 76 Tex. L. Rev. 3 at 553-593 < http://ir.lawnet.fordham.edu/faculty_scholarship/42/>.
[359] WIPO Copyright Treaty, *adopted* Dec. 20, 1996, WIPO Doc. CRNRIDC/94, Articles 11–12; U.S. Digital Millennium Copyright Act 1998 (DCMS), Pub. L. No. 105-304, 112 Stat. 2860 (Oct. 28, 1998); Supra n. 10 (EU InfoSoc Directive, article 6&7); and *Copyright Act*, R.S.C., 1985, c. C-42 , s.41.

only affects the distribution right of copyright holders. In other words, it does not directly affect reproduction, public performance and use rights of holders.

Does the above principle work for computer software? The answer is "yes" and "no". Let us examine the European approach first to address the "yes" answer. Article 4(2) of the Directive explicitly allows the application of software exhaustion if the copy of the computer program is placed in the community market by the right holder or with his consent. However, the application of this section in connection with software licenses and their effect remained a contentious issue among scholars and courts. A case in point is the 2012 *UsedSoft* European Court of Justice Case. As has been explained in preceding sections, the court applied Article 4(2) of the directive to used software licenses. Accordingly, Article 4(2) applies when the users download computer programs online with the consent of the right holder. This acquirer again can further distribute and the principle of exhaustion applies on one condition- when the first acquirer erases the program or no longer uses it.[360] The tricky part comes in proving this last condition. Though the directive and the highest court's decision in EU seem to weaken[361] the distribution rights of software holders, some believe that it is impossible for right holders to prove this fact.[362] Based on UsedSoft, permanent licensing of software upon receiving commensurate fee amounts to sale. Such is not the case in other copyrightable works.

Contrary to the EU approach, the U.S. gives software right holders strong rights to control the distribution of software. That means, the principle of first sale seldom applies in the U.S. software market. Also, companies rarely sell software as their typical market model is licensing. This being the practice, the U.S. *Copyright Act* in s. 109 (a) specifically addresses the principle of exhaustion. According to this section, lawful acquirers are entitled to sell or dispose of works provided they meet the conditions. Logic dictates acquirers of computer programs have similar rights as programs have been regarded as copyrightable in the 1980 amendment. A further

[360] *Supra* note 289, par 70 and 78
[361] Lazaros G. Grigoriadis, "Exhaustion and Software Resale Rights in Light of Recent EU Case Law", online: 2014) 5 J. Int'l Media & Entertainment Law 1 at 111 <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2403554>
[362] See Ole-Andreas Rognstad , "Legally Flawed but Politically Sound? Digital Exhaustion of Copyright in Europe after UsedSoft" online: (2014) 1 Oslo L Rev < https://www.journals.uio.no/index.php/oslawreview/article/view/977

amendment undertaken in 1990 restricted the commercial rental, lease, and lending of computer programs.[363]

3.5 Access rights of the public to technological outputs

Gigantic software companies like Microsoft and Apple obtain thousands of software patents and build their portfolios. Such monopolies on software algorithm affect the software market competition. It will be very difficult for small software companies to use those algorithms and join markets.[364] This monopoly leaves the public with no choice than living the "my way or the highway" approach of software companies. Be this as it may, there are limited practices which seem to consider the interests of new entrants and consumers of software: the doctrine of reverse engineering and open source and free software movements.

3.5.1 Reverse engineering and the public interest

This is an attempt made by the existing system to address the interests of consumers and new entrants. Reverse engineering is breaking down the computer readable object code to human readable source code. Such activity may be done for numerous purposes. Achieving academic and research goals can be regarded as one reason for reverse engineering: to show students or researchers[365] how software codes are written ("the inner workings"[366]). Secondly, by reverse engineering programmers may help resolve software problems, as software is full of bugs. We need to study the program to fix the problem and allow for the improvement of works. For Lande and Sobin, the activity of reverse engineering could be undertaken for three purposes: to create identical software products, to create equivalence or to build interoperable software.[367]

Reverse engineering can be called by many names. The one used under EU law is decompilation. The precise parameter between these terms is unclear. This is a situation in which someone tries to derive the source code from an analysis of object code.[368] They may also try to derive source

[363] *Computer Software Rental Amendments Act*, 17 U.S.C. § 109(b)(1)(a)) (1990).
[364] *Supra* note 215.
[365] Many copyright laws have a fair use or dealing exception which includes the lawful use of works for research or private use.
[366] Robert H. Lande and Sturgis M. Sobin, "Reverse Engineering of Computer Software and U.S. Anti-trust Law", (1996) 9 :2 Harv JL & Tech 238 at 240.
[367] *Ibid*, at 241. The interoperability purpose has been widely accepted by cases and legislations as a main ground for reverse engineering computer software.
[368]Gary R. Ignatin, "Let the Hackers Hack: Allowing the Reverse Engineering of Copyrighted Computer

code from the analysis of the function of software. Hollaar states: "the key decisions on the legality of reverse engineering have dealt with disassembly: taking the publicly-available object code and attempting to reconstruct the original source code to learn how the program works."[369] The point here is that these days there are zillions of software codes affecting our lives. Software rights holders and firms want to fence off their respective programs so that they obtain the optimal economic value, and they oppose the concept of reverse engineering[370]. On the other side, scientists and programmers spend much of their time in studying the already available codes. The public needs competition in the software market as well as quality, compatible and interoperable software products. Hunda also extensively argued for a public interest defense of reverse engineering computer software.[371]The open source and free software groups largely advocate for open system software -the free and wider distribution of software, hence, claiming for the permission of reverse engineering.

Reverse engineering or decompilation may be done to ascertain the underlying ideas and interface specifications of that specific program. The act of disassembling programs and reverse engineering them may constitute a copyright infringement, at least in the existing copyright protection regime. But, one can also argue that such acts of reverse engineering can be supported by the fair dealing[372] exceptions of copyright law. The problem arises in defining fair dealing concept in software, and in determining how much decompilation falls under the fair dealing exception.

Basically, computer programs are protected in their source code and object code form. Source codes can be read and easily understood by human beings. If the source code is complicated for ordinary people, programmers can still read and understand it. In practice, programmers will not make publicly available their source code. Buyers/licensees will only access the object code form of the program, which is understood only by computers. Accordingly, users and some free software advocates claim the permission of de-compilations and reverse engineering.

Programs to Achieve Compatibility" (1992) 140 U. Penn. L. Rev. 1999 at 2000 (for him reverse engineering is transforming the ones and zeros in to a form that is readable by humans).

[369] Supra note 233at 110.

[370] *Supra* note 271 at 238.

[371] *Supra* note 272 at 645-647.

[372] Fair dealing is the grand exception in many copyright laws including international copyright treaties. It is called fair use doctrine in U.S..

Software reverse engineering mostly raises trade secret and copyright violation issues. Patent infringement is unlikely in this regard, as the concept of a patent requires complete disclosure. In some countries such as the U.S., patent applicants need to explain the best mode of doing the product or process [software]. Such requirement is missing in trade secret and copyright scenarios.

In the U.S., the Supreme Court defined reverse engineering as *"a fair and honest means of starting with the known product and working backward to divine the process which aided in the development of manufacture."*[373] Though they approach reverse engineering broadly, some authors believe the Supreme Court's pronouncement is the standard definition.[374]

The other major case concerning software reverse engineering in the U.S. is *Sega Enterprises Ltd. v. Accolade, Inc.*[375] The case involves reverse engineering and the fair use doctrine in software copyrights. Sega sued Accolade in the District Court claiming the defendant violated its right by disassembling Sega's software codes. Accolade, on the other hand, argued its act fell under the fair use exception. The District Court ruled in favor of Sega, and the defendant appealed to the court of appeal for the Federal Circuit. The later court reversed the District Court's decision and allowed the disassembly based on the fair use doctrine. The court stated that Accolade's act of disassembly was intended to ensure compatibility.[376]

In the EU, the directive allows decompilation under limited grounds: for the sake of creating interoperability of an independently created computer program with other programs.[377] Recital 10 of the directive defines interoperability as *"the ability to exchange information and mutually*

---

[373] *Kewanee Oil Co. v. Bicron Corp., 416 U.S. 470, 476 (1974).*
[374] Pamela Samuelson and Suzanne Scotchmer, "The Law and Economics of Reverse Engineering", (2001) 111 Yale L.J. 1575 at 1577 (for them reverse engineering is *"the process of extracting know-how or knowledge from a human-made artifact"*).
[375] *Sega Enterprises Ltd. v. Accolade, Inc.*, 977 F.2d 1510 (9th Cir. 1992) (This is just one example). There other court cases that reach on similar conclusion with *Sega.* See, for instance, *Atari games Corp. v. Nintendo of America, Inc*.,975 F.2d 832 (Fed Cr. 1992)
[376] *Ibid* at1518, 1522 (It particularly stated the disassembly is undertaken "*solely in order to discover the functional requirements for compatibility with the Genesis console-aspects of Sega's programs that are not protected by copyright.").*
[377] Supra note 12, article 6; supra note 336 at 1609 (The process of disassembling helps to obtain information necessary to develop a program that will interoperate with the decompiled or disassembled program); Kathleen Gilbert-Macmillan, "Intellectual Property Law for Reverse Engineering Computer Programs in the European Community", (1993) 9 Santa Clara High Tech. L.J. 247 at 248, 251.

to use the information which has been exchanged." Sub-Article 3 of Article 5 seems to allow very limited forms of reverse engineering. If we see the Sub-Article as a whole, it is very difficult to get a firm grip on it. But the argument that this Sub-article is about reverse engineering may raise a problem for a carefully worded concept of decompilation under Article 6.

What Article 6 does is allow decompilation only for the purpose of achieving interoperability; *i.e.*, interoperability with some other programs. The rights to reverse engineer set out in Article 6 are very restricted. This restriction is also underlined in recital 15 of the preamble. One such restriction is the requirement of necessity. According to Article 6(1) (b), decompilation is also not allowed if the information necessary to achieve interoperability has previously been readily available.

Reverse engineering copyright software codes faces other challenges. As have been raised in the preceding sections, copyright holders use not only copyright laws to ensure protections over their works. They also use Technological Protection Mechanisms (TPMs) to enforce digital copyrightable works. Those TPMs or otherwise referred to as DRMS are protected under the law. These laws indirectly outline the self-regulatory nature of technologies. Just to mention few, the WCT under Articles 11 and 12, the EU information Society Directive under Articles 6 and 7, and the US Digital Millennium Copyright Act lay down safeguards to TPMs/DRMS. By doing so the law empower right holders to control access, copy, and protect the authenticity of their works. Software is a very good example of digital software work. Accordingly, technologies which are developed to circumvent these mechanisms (e.g. in the form of reverse engineering/ decompilation) should be outlawed based on the above laws. This, in other words, makes very difficult or impossible to reverse engineer software codes. For instance, companies developing video game software, using DRMS, make difficult to play certain games outside the specified regions.

3.5.2 Free and Open Source software movements favoring the interest of the public

In the foregoing sections, we discussed proprietary software. Software companies, as described above, use all possible mechanisms to make optimal profits. They particularly use trade secret

and copyright so that source code lines (recipe) remain hidden from the reach of small entrants and software users.[378] Only software in its object form (machine readable) is made available to the market. The open source approach helps balance the IBM- and Apple- like monopolist's strategy. Open source software intends, among other things, to make freely and publicly available source codes.[379] Sometimes, the free and open source approach is not only in the interest of users' rights. Also, developers of software in the computing industry, such as bioinformatics scientists, are increasingly requesting the expansion of open source distribution with their underlying software lines.

The Free and open source communities are not merely opposing proprietary software. They have shown us a significant contribution to the public and small software industries. They developed various software forms, most of which serve as an alternative to proprietary software. The notable free and open source software developed so far include *GNU* operating system, Apache Open office, Google Chrome and Firefox, Pdf Creator, Mplayer and VLC media player etc. The societal benefits of this software and their detriment to proprietary software industry are undeniable. For instance, Google Chrome and Firefox replace Internet Explorer, and Apache open office totally replaces MS office.[380]

Different free and open source foundations and organizations have been established. The Free Software Foundation (FSF) launched by Richard Stallman[381] is one successful organization. It started in developing the GNU operating system, and today it is contributing significantly to the free and open source software community. It played an even greater role in the adoption of legislation concerning computer software (for example, it played a significant role in the rejection of EU software patent directive). To ensure the further free distribution of software, the foundation developed the GNU "Copyleft public license" system. The license prohibits

---

[378] David S. Evans & Anne Layne-Farrar, "Software Patents and Open Source: The Battle over Intellectual Property Rights, (2004) 9:10 Va JL & TECH at 3; *Ronald J. Mann*, "Commercializing Open Source Software: Do Property Rights Still Matter?" (2006) 20: 1 Harv JL & Tech at 23 [however, some companies such as Microsoft began to allow access to certain source code parts.]; Jonathan Zittrain, "Normative Principles for Evaluating Free and Proprietary Software", (2004) 71 U *Chicago L Rev* (software companies hide source code recipe so that the public cannot view it, and other programmer s do not develop a new and improved software).

[379] *Ibid* (David S. Evans & Anne Layne-Farrar).

[380] For further understanding of the societal benefit of free and open source software, see Richard Stallman and Lawrence Lessig, Free Software, Free Society: Selected Essays of Richard M. Stallman., 2nd ed. (Boston: Createspace, 2009).

[381] He left his job from MIT to pursue his free software project. Stallman decided to replace 'Unix' with other operating system. Accordingly, he developed *GNU* (*GNU* is not *UNIX*) operating system.

developers of derivative software from placing restrictions while distributing their work. This way, the modified versions of software remain free. Without this license system, some programmers would have changed their improved works to the proprietary software type. The other notable organization is the Apache Software Foundation (ASF). Akin to FSF, ASF is also assisting the open source software projects. Correspondingly, the GNU-like public license is also in place in the ASF. The Electronic Frontier Foundation (EFF) is the other category that is hugely supporting the open source movement. It defends digital rights of consumers and challenges any attempt to restrict civil liberties in the digital environment. With regard to software, it has launched a lawsuit to challenge the constitutionality of the US DMCA[382], among others. In the suit, the Foundation believes the expansion of software into a modern product is effectively locking down everything. It then argues "the anti-circumvention section of DMCA threatens fair use, impends competition, and innovation, and chills free expression and scientific research."[383]

One thing should be clear here. In the free and open source software system, the main issue is access, particularly, to the software recipe (source code lines). It should not be related with "free" as "free in price". Improved or modified versions could be distributed freely (free of price) or in the form of sale. As the ultimate aim is ensuring all users have the freedom to access the software recipe, Copyleft public licenses place restrictions on subsequent developers: an obligation to leave their works source code accessible. In explaining the significance of open and free software to the public, Lessig has said:

> "Open source and free software give consumers and the public something more than proprietary software does: the ability to tinker and modify. Such software gives the public the benefit of the information contained within the code."[384]

Besides, the approach seems to be consistent with the traditional notions of innovation and creation[385], other than their importance to software consumers and the public.

---

[382] Digital Millennium Copyright Act, online: The Electronic Frontier Foundation <https://www.eff.org/issues/dmca>.
[383] *Ibid*.
[384] Lawrence Lessig, "Open Source Baselines: Compared to what?" in Frederick M. Abbot, Thomas Cottier and Francis Gurry, *International Intellectual Property in an Integrated World of Economy,* (New York: Wolters Kluwer, 2015) 692 at 693.
[385] *Ibid*.

To summarize, the importance of open and free software movement to the consumer and small software companies is unquestionable. Consumers will be able to use and access software application for no cost. Also, software developers could freely access the openly available software algorithms and study the inner working of software codes. This creates an opportunity for small entrants to develop a better software products. However, the existing system let alone to regulate how these free and open source software organize develop and distribute their software, it does not even mention of the existence of these categories of software.

CHAPTER FOUR

CONCLUDING REMARKS AND RECOMMENDATIONS

4.1 Concluding remarks

Today we call the era the information age, computer age or digital age. Whatever explanation someone uses, the era is characterized by a huge explosion of technological innovations. Society as a whole is now regarded as a knowledge-based society. Information is becoming a currency and the economy is highly dependent on computer related technologies. Computer software is the most important element of computing technology that has significantly enhanced the so-called information economy. It is difficult to find a life without the direct or indirect involvement of software technologies. Hence, the increasing importance of computer software demands careful regulation. Accordingly, many efforts have been made to put in place regulatory frameworks for the software industry but, to date, the industry remains the subject of fierce academic discourse and court litigation.

One thing less controversial about software is its "intellectual object" nature. Although there is no consensus as to the proper form of intellectual property rights software should enjoy, there is no controversy regarding the need to have some sort of protection. In approaching the existing framework for intellectual property protection for computer software, we observe many concerns. For instance, it is not even clear what computer software is. The terms "computer program" and "software" have been used interchangeably in much literature and in this work. However, it is necessary to make a clear distinction between the two. The use of the subject "computer" to software and programs is equally confusing. Software is not only used in a computer. Other devices such as mobile phones, televisions *etc.* use software. Generally, we can define software as a logical set of instructions that help a computing or other device perform a specific function that produce a certain result. In other words, the computer only functions and produces a result when the system software (OS) and the purpose- specific application software are installed.

There are some attempts to define the subject in certain legislations, but that does not give relief. The approach in the U.S. and Canada seems consistent. They define software in terms of what it is and its function. However, the scenario is different when one sees the approaches taken by the

two most important international instruments and the EU directive. In the later instruments, software is defined in terms of its scope. One unique element added in the EU directive is the inclusion of preparatory material as a defining element. Though non-binding, the WIPO model provision seems better and elaborative. For instance, "computer program" is a subset of computer software as per the Model provision.

Hence, existing laws do not properly address the definitional issues. It would be helpful if those definitions give clarification to different forms of software such as application software and operating system software. What exactly constitutes source code, object code, interfaces, preparatory documents, chips, and related notions should have been addressed either by legislation or judicial pronouncement. Software development involves the writing of codes, the application of different programming languages, etc. So, these elements and technical procedures should be considered in clarifying software through definition or delimitation of scope.

The second point concerns the justification of software regulation. Why we do regulate software at all? Does software fit into the general theoretical justification of property rights? As has been pointed out in the foregoing sections, software is unique and is a very complex legal and technological concept; thus, it requires a contextual approach in justifying its scope and terms of protection. Borrowing certain principles from general and intellectual property laws is not difficult. However, a carefully studied and contextual approach in justifying the legal protection of computer software is imperative.

Even if there is ambiguity and uncertainty as regards many issues pertaining to computer software, that does not signify an absence of law on the subject. There are various forms of protection. If there is a single intellectual property right which is born under a lucky star, that is software. Some, without considering other forms of protections, regard computer programs as "*the golden child"*[386]in the realm of copyright. It gets the protection of almost all traditional intellectual property rights.

For the most part, the U.S. is considered the leading country in the software industry. Software developers claimed software protection in the early 1960s. The copyright office began

[386] Christina M Reger, "Let's Swap Copyright for Code: The Computer Software Disclosure Dichotomy", (2004) 24 Loy LA Ent LR 215 at 217 (the main reason for this is software copyright required *limited disclosure* unlike other copyrightable literary works, nonetheless gets similar protection).

registering software copyrights before the *Copyright Act* was amended. During that time, computer software was regarded as books, falling in the class of literary works. The then-famous copyright law at the international level, Berne Convention for the Protection of Literary and Artistic Works, did not even mention the term computer software (even in its 1979 version). However, this was not surprising in the U.S. as the U.S. has not signed this document. A comprehensive revision of the Copyright Act has been undertaken and copyrighting software remained one controversial issue. As Congress believed in a sober examination of the matter, it established a commission (CONTU) to come up with recommendations. Based on CONTU's recommendations, the Copyright Act was amended in 1980 and included two sections about computer software.[387] For some, this seemed the end of all the uncertainties and ambiguities. However, the courts continued the tortuous battle of delineating the scope of software copyrights and establishing tests for copyrightability. Markedly significant cases worthy of mentioning are *Williams Electronics, Inc. v. Artic International, Inc., Apple Computer, Inc. v. Franklin Computer Corp, Whelan Assocs., Inc. v. Jaslow Dental Laboratory, Inc., Lotus Dev. Corp. v. Paperback Software and Mosaic Software, Computer Associates International Inc. v. Altai Inc., Lotus Development Corp. v. Borland International, Inc.,* and the recent *Oracle America, Inc. v. Google, Inc.* case.

Canada's 1988 Copyright Amendment Act regarded computer software as a literary work. Though not routine as in the U.S. approach, Canadian courts have held computer software copyrightable in many cases, most notably *Apple Computer Inc. v. Mackintosh Computers Ltd.* [388] Firstly, it is a case which involved all the three levels of courts. Most importantly, it is the case that marked for the first time that computer software in its source and object code is regarded as copyrightable.

The approach in the EU is somewhat different. Though the EU is 13 years late in a legislative rule on software, there is at least a harmonizing directive that applies to all member states in the union. The directive is particularly interesting for the following reasons. Firstly, it has a referral provision to the Berne convention for the Protection of Literary and Artistic Works (which is regarded by many as the constitution for copyright). Secondly, it adopts a very general and broad

[387] supra note 72, §101 and §117.
[388] *Supra* note 269.

meaning for computer programs. Its broader definition is manifested, for instance, by the inclusion of "preparatory design material" as one protectable work. The directive also attempted to articulate the economic rights of software copyright holders. Another key point the directive introduced is the notion of decompilation (often used synonymously with reverse engineering) and interoperability issue. In similar fashion with the U.S. and Canadian courts, courts in Europe have entertained many software copyright cases. Some of the decisions of the highest court of the EU are even found having a strong effect on the existing software copyright discourse. The most praised decision, *SAS Institute Inc. v World Programming Ltd,* excluded programming language and functionality of programs from the scope of software copyright. Furthermore, the court in the *UsedSoft GmbH v Oracle International Corp* case elaborated the doctrine of exhaustion (first-sale) in the software context. The *UsedSoft* court pronounced that use right for unlimited period is a sale. Hence, the acquirer (the licensee) is entitled to further transfer the program. In October 2016, the court again handed down another software case, between *Mr. Aleksandrs Ranks and Mr. Jurijs Vasiļevičs v. Department for the Prosecution of Economic and Financial Offences, Latvia and Microsoft Corp.*[389] The CJEU interpreted Article 4(a) and (c) and Article 5(1) and (2) of the software directive. This shows the significant development of software court cases.

At this time, it is clear that there is no law that permits the patentability of computer software. In some jurisdictions, such as in Europe, computer programs are excluded from the reach of patent law. Section 52 of the European patent convention is an excellent example in this regard as it considers *pure software* inventions unpatentable. The convention only excludes *computer programs as such,* leaving many unsolved questions as to programs other than 'programs as such'. Furthermore, there was an unsuccessful attempt to issue a directive for computer software.[390]

The European patent office has developed interesting jurisprudence. It coined the tests of a *technicality* in granting patents to software products. The test qualifies the blanket exclusion of software patents under Article 52 (3) of EPC. Accordingly, patent could be available if the claim involves technicality feature- provides technical solution to technical problem. The office has

[389] Mr. Aleksandrs Ranks and Mr Jurijs Vasiļevičs v. Department for the Prosecution of Economic and Financial Offences, Latvia and Microsoft Corp. Case C-166/15,
[390] *Supra* note 20.

also recently amended its guideline[391] for examination and uses similar terminology. Although the EPC excludes computer programs *per se*, evidence shows that there is some kind of leniency at the patent office level, and thousands of software patents have been issued since 1978.

In the U.S., although Congress decided that copyright law best suits computer software, hundreds of thousands of software patents have been granted by the U.S. Patent and Trademark Office (USPTO) and the number of software patents has grown exponentially. This, obviously, shows the anomaly of protecting literary works with patent laws.

Unlike the EU and Canada, the U.S. has a lot of case laws concerning computer software. The District Courts, Federal Circuits and Supreme Court have decided many software patent cases since the beginning of the 1970s. At this time, one cannot know conclusively the position of courts as regards the patentability of computer software. However, there is ample evidence for and against software patents. Since *Benson,* courts continued to develop more than ten tests of patentability of software. In *Benson,* the court used the preemption of claims to a mathematical algorithm in determining patentability. The *Parker* court used the contribution of claimed process in the article's nature or state. In 1980 the *Diehr* court granted software based on the test that the claimed process involves the transformation of an article and disregarded the blanket exclusion of mathematical formula and algorithm. Two years later, the Federal circuit came up with the *Freeman-Walter- Abele Test.* Two-step tests have been developed: if the claim recites a mathematical algorithm, and whether the claim as a whole is no more than the algorithm itself: if our answer is positive, then the claim is non-statutory subject matter. In 1994, the *Alappat* court adopted the *useful, concrete and tangible test.* Controversy continued and in 2008, the *Bilski* court coined another test – the machine and transformation test (a test that allows patentability if the claimed machine/process ties with a particular apparatus or transforms a particular article into a different state or thing). The Supreme Court rejected this test and stated the ultimate determination must be whether the subject matter is a law of nature, physical phenomena or abstract idea – arguing these categories of subject matters are absolutely not patentable. The 2015 *Alice* court again adopted other two-step tests. First, the court should determine if "the claims at issue are directed to one of those patent-ineligible concepts*".* Secondly, "If so, the [court should] then ask, what else is there in the claims before us?" In the

[391] *Supra* note 207.

latter step, the court is asking if there is an *inventive concept* that amounts to significantly more than the patent ineligible concept itself.

Similar to the U.S., there is no express exclusionary section of patenting computer software in the *Patent Act* of Canada. However, we see some scholarly discourses on the matter – from the interpretation of laws to exclude software patentability to the existence of legislation and practices allowing patents for software. Some position the Canadian approach between the U.S. and EU. What is clear is that certain software and business method patents have been granted in Canada (e.g. the recent *Amazon* patents), and there is a guideline from 2007 (amended recently)[392] that allows patenting software so long as the claim is integrated with another patent eligible subject matter.

As stated above, patent and copyright are not the only forms of intellectual property mechanisms for computer software. A trade secret is known for protecting the internal working and design of software source code. Unlike copyright and patent, trade secret protection for the hidden part of software has not been particularly controversial. It has always protected source code, the inner working of software – things we cannot see. Recent proposals even seek for the possibility of protecting the *revealed* aspects of software.

All these protections, though favored by the software industry, disregard the very purpose of intellectual property rights – the provision of limited protection to intellectual works for the *public good*. There is nothing good for the public by over-protecting an intellectual good which is so important to the everyday life of the public. It is not, therefore, hyperbole to claim the existing system is a double-edged sword: it highly benefits gigantic hardware and software companies and ignores the general interest of the public. Currently, one can only see a very limited scenario in terms of the existing approach's consideration towards the public interest: the limited allowability of reverse engineering software and the Open Source and Free Software movements. In particular, the Open source movement seems to significantly address the access related issues of software. However, there is no recognition of that movement in the abovementioned intellectual property laws.

[392]Supra note 190 at105.

Studies show that computer software is a unique form of intellectual good.[393] It is unique in the sense that it is not a mere book like a work as understood by the 1964 U.S. copyright office and subsequent laws and bodies. Software is also a complex technological innovation that affects every aspect of human life. If we dissect and analyze it, we see a whole lot of complex legal and technological notions: from source and object code to the look and feels aspects; from operating system to application software; from chips, diskette, and memory to programming language; from free and open source to proprietary software, etc.

It is also beyond doubt that traditional forms of intellectual property rights are not suitable to software. The main reason, among others, is that software is unique, complex and omnipresent and is an essential element of so much evolving technology. Many scholars have argued the inapplicability of existing system as it is and proffered many alternatives. Some such as Pamela Samuelsson challenge the copyright protection of computer software. Others oppose software patents. Professor Peter Menell proposes the option of protecting some part of software (operating system) with patent and copyright for the remainder. This research proposes a special form of protection for computer software.

4.2 Recommendations

When one talks about computer software, emphasis should be given to its nature, especially to its unique traits. The basic of these are the technological, complex (except for programmers) and omnipresent nature. If something is unique, that means we have to approach it in context. Hence, the main remedy to rectify the existing blanket copyright or patent or trade secret or trade dress or other forms of protection of software is to devise a special law for software. There is even no need to categorize software as a patentable or copyrightable subject matter. It is sufficient if it enjoys the necessary *(special) or standalone* protection. This way we are avoiding, firstly, the age-old litigation of attempting to determine whether software is an invention or literary creative work. Additionally, we will have up-to-date and fully-fledged law that addresses many technical issues from definition to scope and tests of infringement/encroachment. Above all, the new

[393] For instance, see supra note 86. David Hayes argues software has seven unique characteristics. Software is inherently functional, embodies multiple types of creativity, its evolution is often incremental, it is increasingly short-lived, software development methodology has evolved, it exists in different markets, and software has many different distribution and use architectures.

special law will take into consideration the real impact of software on society and its role in the digital economy.

If one examines the existing legal regime, it is clear that the system ill-suits the interests of consumers. Proponents of free software claim for software to be free. Of course, all consumers want the free and open distribution of software products. A new entrant, too, needs some kind of access to software innovation so that they can build up their own initiatives. However, the intellectual object nature of software is less controversial. Software developing companies, though they benefit from the wealth of freely available information, invest so much time, labor and skill, and money in writing software codes. Logic dictates these companies should be able to recover all the costs incurred and allowed to earn an appropriate (fair) profit. The problematic question is "what is appropriate?" Obviously, the existing system is not appropriate. Allowing a 20 years patent protection, and after expiry extending that protection to fifty[394] or seventy[395] years is not by any means appropriate. Hence, recognition should be given to programmers of software. But the reward we give for these programmers in the form of intellectual property rights should also take into consideration of the reality, and the interests of various stakeholders. For instance, attention should be given to the interest of the consumer, programmers and new entrants.

This new regulation does not need to be called patent or software or copyright or even *sui generis* protection for computer software. It is enough to enact a statute or an act for software. The point is that it is not adequate in having a separate legal document for computer software. EU has that. However, what EU has is a separate *copyright* directive for computer software. WIPO proposed the *sui generis* approach in the late 1970s. It was a better proposal, but the content is no better than the 11Articles length of EU software directive. The WIPO *sui generis* model provisions proposed in early 1980s only addresses few issues. The only substantive elements the model provision had introduced were definition of terminologies, duration of protection and the impositions of general prohibitions. According to this proposal, computer software includes computer program, program description and supporting document. Though it provides a definition to computer programs like the U.S. and Canada Copyright *Acts*, the

[394] *Supra* note 73, s.6.
[395] Directive 2006/116/EC of the European Parliament and of the Council of 12 December 2006 on the Term of Protection of Copyright and Certain Related Rights, articles 1(3)(6), 2 (2); *supra* note 72, s.302.

definition given to program description and supporting material is not clear. Furthermore, the model provision does not take into context these days billions of software applications used in smart devices. Even copyright laws have been revised to meet the needs of the information age. WIPO's adoption of copyright treaty is one such example. Hence, the model provision though it introduced the notion of regulating software with special law, it did not solve todays software related disputes. For instance, the model provisions let alone to address the look and feel aspects of software it does not even mention of the object and source code elements of software.

The EU directive is adopted a decade after the WIPO proposed the model provisions, and it still does not address central software protection issues. The directive is nothing, but a detailed version of copyright law for software. For instance, whether the directive protects the literal elements of software is not clear.

Perhaps regulating a specific subject matter of intellectual property rights is not that uncommon. In Europe, database is one subject matter that is being governed by the special directive.[396] This directive gives copyright and *sui generis* protections to databases. In the U.S., Semiconductor Chips have been regulated by a special *Act*[397] since the 1980s.

As has been noted in the CONTU report, the main concern for protecting programs was avoiding unauthorized copying of computer software.[398] This recommendation shares that concern. Nonetheless, avoiding illegal copying does not mean we must overprotect programs so that the system unduly benefits the computing industry at the expense of the main purpose of IP laws and other stakeholders.

As regards the structure of the recommended law, in the form of preamble or recital, mention should be made as to what is the main basis for software protection. It may be difficult to find a single justifying theory for computer software regulation. However, it is not appropriate just to treat software like other works or innovation. It all goes with the unique and omnipresent nature of computer software. The utilitarian justification seems the main basis for software protection. All things considered, our justification should not encourage free riders to copy others' ideas.

[396] Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the Legal Protection of Databases.
[397] 17 U.S.C. §§ 901-914 (1988).
[398] Supra note 233 at 60.

Ideas are expensive to produce and easy to copy.[399] The problem of piracy – illegal reproduction and distribution – could be serious in digital works such as software. Hence, the justification should encourage new innovations and creativities.

The content of this specific *standalone* legislation should include, among other things the following points. Firstly, the proposed statute should provide an up-to-date definition for computer software. The existing legislations does not take us far as regards clarification of what a computer program is. Stipulating a clearer and concise meaning for technical terms such as software is not an easy task. However, an attempt to stipulate a binding definition for those terms will help to avoid possible confusions. In doing so, this standalone statute should make a clear distinction among terminologies related to computer programs, such as "software", "computer software", "source code", "object code", "programming language", "programmer", "developer", "interfaces", "look and feel" etc. Furthermore, it should address if software only applies to physical or tangible devices. This is because today internet is being regarded as a machine.

Secondly, the statute should delimit its scope of application. In this part, the legislation will address what elements of software and computing technologies fall under the realm of the special law. In connection to this, this law should identify excluded subject matters. This way, we can reduce time and costs of courts and other concerned bodies. For instance, it could clarify the status of object code, source code, preparatory materials, supporting documents and other non-literal elements.

Thirdly, the statute should explicitly spell out specific rights or privileges developers of computer programs have. At this level, it could be very difficult what rights should this specific law entitles software developers. The law should identify parties involved in developing software.[400] It then should delimit their respective rights and entitlements. The right to store programs in any medium for use or distribution may constitute examples of substantive rights. Then, these rights could be further defined to meet the real software market. For instance, the distribution part may form sale, license, hire or lease.

---

[399] Barry Sookman, Steven Mason, and Carys Craig, *Copyright Cases and commentary on the Canadian and International Law*, 2nd ed., (Toronto, Canada: Carswell, 2013) at 11

[400] At times, individual software programmers could write software codes under the supervision or employment of someone else. Hence, their relationship between those parties should spelled out.

The EU software directive tries to stipulate substantive rights of right holders. The directive, however, only reiterates basic copyright entitlements of authors. It does not even say anything about moral rights of authors, as EU copyright law bestows on authors both economic and moral rights. It is essential for the recommended law to address what the right holder is permitted to do and the *extent* of that entitlement. That could include delimiting the duration of protection, and issues of transfer (e.g. assignment in the form of sale or license *etc*.). The extent of the right and duration should enable holders recoup the cost of developing programs and spur innovation. Additionally, the law should address issues of adaptation, translation or other ways of modifications. Likewise, it is important if this special law clarifies the possibility and conditions of reverse engineering computer programs. A related issue is the notion of interoperability of software.

Fourthly, the law should have a clause on free and open source software. There is no a single provision on the existing regime regarding free and open source software. Currently, they are functioning based on contract and public licensing mechanisms. Hence, the new law should recognize the reality and include regulatory sections for free and open source software. This law assists the existing public licensing mechanisms that free and open source software movements use. The law may guide how consumers use, copy, study and reverse engineer free and open source software.

The most important part relates to infringement. As there is no clear test for infringement of software patents and copyrights, courts are trying to develop different criteria. Therefore, the recommended law by identifying criteria/tests[401] of infringements could ease settlements of litigations. We may have two types of tests. Firstly, it is important to stipulate criteria for software protection. As we have novelty, inventive step and utility criteria for patents, we need to have a specific test for software protection. The second test concerns infringement of protected software. In connection to this, functional similarity could serve as one criteria to determine infringement of right. Last but not least, this standalone legislation should encompass administrative rules. Determination of substantive rights is not sufficient. Some procedural rules, too, should form part of this special law for computer software. In this section, we could address

[401] As has been discussed software is a complex legal and technological notion. Hence, it is not easy to recommend a clear test of infringement. Further examination of the matter by specialists is indispensable in defining the scope of protection and determining test of protection or infringement.

questions such as how to acquire and enforce rights. Should the acquisition of right be automatic or does it require some sort of examination and registration? Also, it could define adjudicative and other enforcement entities.

BIBLIOGRAPHY

LEGISLATION

Agreement on Trade-Related Aspects of Intellectual Property Rights, Annex 1C to the *Final Act and Agreement Establishing the World Trade Organization*, December 15, 1993, 33 I.L.M. 76 (WTO). *General Agreement on Tariffs and Trade,* Uruguay Round (including GATT 1994), Marrakesh, April, 1994.

Berne Convention for the Protection of Literary and Artistic Works, September 9, 1986, *Can T.S.* 1948 No. 22. 828 U.N.T.S. 221, revised most recently by *Paris Act relating to the Berne Convention,* July 24, 1971,

Canadian Patent Act, R.S.C. 1985, c P-4

Computer Software Rental Amendments Act, 17 U.S.C. § 109(b)(1)(a)) (1990).

Constitution Act,1867 (UK), 30 & 31 Vict, c 3, reprinted in RSC 1985, App II, No 5

Convention on the Grant of European Patents (European Patent Convention of 5 October 1973 as revised by the Act revising article 63 EPC of 17 December 1991 and the Act revising the EPC of 29 November 2000.

Copyright Act, R.S.C., 1985, c. C-42.

Council Directive on the Legal Protection of Computer Programs, No. 91/250, O.J. L 122/42 (1991).

Council of Ministers, Common Position Paper, Art. 1, At 7 (Dec. 14, 1990).

Criminal Code, R.S.C.1985, c. C-46

Directive 2001/29/EC Of The European Parliament and of the Council of 22 May 2001 on the harmonization of certain aspects of copyright and related rights in the information society.

Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases

EPO, Guidelines for Examination in the European Patent Office, Nov 2016, ISBN 978-3-89605-158-5< http://www.epo.org/law-practice/legal-texts/guidelines.html >.

Guidelines for Examination in the European Patent Office November 2016.

Model Provisions on the Protection of Computer Software, 12 Indus. PROP.: Monthly REV. WIPO 259-73 (1977).

Omnibus Trade and Competitiveness Act, 19 U.S.C. § 2242, 2411-2420 (West Supp. 1990).

Patent Act, 35 U. S. C. §101.

Procedure 2002/0047/COD COM (2002) 92: Proposal for a Directive of the European Parliament and of the Council on the patentability of computer-implemented inventions.

Proclamation No.410/2004 *Copyright and Neighboring Rights Protection Proclamation.*

Proposal for a Council Directive on the Legal Protection of Computer Programs, O.J. C 91/4 **(**1989); Amended Proposal for a Council Directive on the Legal Protection of Computer Programs, O.J. C 320/12 (Oct. 1990).

Proposal for a Directive of the European Parliament and of the Council on the patentability of computer-implemented inventions (2002/C 151 E/05) COM (2002) 92 final — 2002/0047(COD)

Statute of Anne of 1709 (U.K), 8 Anne, c.21.

US Copyright Act, 17 U.S.C. (1976)

U.S. Const. art. I, § 8, cl. 8.

U.S. Digital Millennium Copyright Act 1998 (DCMS), Pub. L. No. 105-304, 112 Stat. 2860 (Oct. 28, 1998

World Intellectual Property Organization Copyright Treaty, December 23, 1996, CRNR/DC/94.

JURISPRUDENCE

Alice Corp. v. CLS Bank Int'l, 134 S. Ct. at 2354

Apple Computer, Inc. v. Franklin Compute Corp, 714 F.2d 1240 (3d Cir. 1983), *rev'g* 545 F. Supp. 812 (E.D. Pa. 1982), 714 F.2d 1240 (3d Cir. 1983), *rev'g* 545 F. Supp. 812 (E.D. Pa. 1982).

Apple Computer Inc. v. Mackintosh Computers Ltd ,10 C.P.R. (3d) 1 (F.C.T.D. 1986); aff'd, 18 C.P.R. (3d) 119 (F.C.A. 1987).

Arrhythmia Research Technology Inc. v. Corazonix Corp , 958 F.2d 1053, 22 USPO2d 1033 (1992).

Bernard L. BILSKI Rand A. Warsaw No. 2007-1130., 545 F.3d 943.

Bilski v. Kappos, 130 S. Ct. 3218, 3225 (2010).

Canada (Attorney General) v. Amazon.com, Inc., [2011] FCA 127.

Canadian Admiral Corporation Ltd. v. Rediffusion Inc, [1954] Ex. CR 382, 20 CPR 75.

CCH Canadian Ltd. v. Law Society of Upper Canada , [2004] 1 S.C.R. 339, 2004 SCC 13.

Diamond v. Diehr, 450 U.S. 175 (198)

Computer Associates International Inc. v. Altai Inc , 75 F.Supp. 544, 20 USPQ2d 1641.

Computer Edge Pty. Ltd. v. *Apple Computer Inc.* (1986) 161 CLR 171

FEIST PUBLICATIONS, INC. v. RURAL TELEPHONE SERVICE CO., 499 U.S. 340 (1991).

Freeman, 573 F.2d 1237 (C.C.P.A. 1978); Walter, 618 F.2d 758 (C.C.P.A. 1980); and Abele, 684 F.2d 902 (C.C.P.A. 1982).

Gottschalk v. Benson, 409 U.S 63 (1972)

IBM v. Spirales Computer Inc , 80 C.P.R. (2d) 187 (1984).

Interstate Circuit, Inc. v. United States ,306 U.S. (1939) at par. 30.

Kewanee Oil Co. v. Bicron Corp., 416 U.S. 470, 476 (1974).

Lotus Dev. Corp. v. Paperback Software and Mosaic Software , 740 F. Supp. 37 (D. Mass. 1990).

Lotus Development Corp. v. Borland International, Inc , 799 F. Supp. 203 (D. Mass. 1992) [Borland I1]; 788 F. Supp. 78 (D. Mass. 1992).

Mayo Collaborative Services, Dba Mayo Medical Laboratories, Et Al. V. Prometheus Laboratories, Inc., 132 S. Ct. 1289 (2012).

Mazer v. Stein, 347 U.S. (1954).

Mr. Aleksandrs Ranks and Mr Jurijs Vasiļevičs v. Department for the Prosecution of Economic and Financial Offences, Latvia and Microsoft Corp. Case C-166/15.

Navitaire Inc v EasyJet Airline Co Ltd [2004] EWHC 1725 (Ch.).

Nova Productions Ltd v Mazooma Games Ltd [2007] EWCA Civ 219.

Oracle America, Inc. v. Google, Inc , 750 F. 3d 1339 (2014).

Parker v. Flook, 437 U.S. 584 (1978).

RDG Inc. v. Dynabec Ltd, 6 C.P.R. (3d) 299 (1985).

SAS Institute Inc. v World Programming Ltd, Case C-406/10.

Sega Enterprises Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992).

Schlumberger Canada Ltd. V Commissioner of Patents , 56, 204(1984).

Spacefile Ltd v. Smart Computing Systems Ltd , 75 C.P.R. (2d) 281 (1983).

*State Street Bank & Trust Co. v. Signature Financial Group, Inc.,*149 F.3d 1368, 47 U.S.P.Q.2d (BNA) 1596 (Fed. Cir. 1998), cert. denied, 119S. Ct. 851 (1999).

Tennessee Eastman Co. v Canada (Commissioner of Patents) (1972), 8 CPR (2d) 202 (SCC).

*Twentieth Century Music Corp. v. Aiken,*422 U.S. 45 L. Ed. 2d 84, 95 S. Ct. 2040 (1975).

Ultramercial, LLC v. Hulu, LLC, 657 f.3d 1323 (Fed. Cir. 2011), reh'g and reh'g en banc denied, No. 2010-1544, 2011 U.S. App. LEXIS 25055 (Fed. Cir. Nov. 18, 2011).

UsedSoft GmbH v Oracle International Corp, Case C-128/11).

United States v. Paramount Pictures, Inc., 334 U.S 131 (1948).

Whelan Assocs. Inc. v. Jaslow Dental Laboratory Inc., 797 F.2d 1222 (3d Cir. 1986) *cert. denied*, 479 U.S. 1031 (1987).

Williams Electronics, Inc. v. Artic International, Inc, 685 F.2d 870 (3d Cir. 1982).

SECONDARY MATERIAL

Aharonian, Greg, "Deconstructing Software Copyright, 30 Years of Bad Logic", (2001) online: Internet Patent News Service < http://www.patenting-art.com/copyprob/softcopy.htm>.

Andrews, Robin, "Copyright Infringement and the Internet: An Economic Analysis of Crime", (2005)11:2 BUJ Sci. & Tech L.

Aplin Tanya. & Davis Jennifer. Intellectual Property Law: Texts, Cases, and Materials, (New York; Oxford University Press, 2009).

Arnold, Richard and Henry Carr, *Computer Software: Legal Protection in, 2nd ed.,(London, UK* : Sweet & Maxwell, 1992).

Azar, Deborah, "A Method to Protect Computer Programs: The Integration of Copyright, Trade Secrets, and Anticircumvention Measures" online: (2008) Utah Law Review 4 1395<http://epubs.utah.edu/index.php/ulr/article/view/135/117>.

Bainbridge, David, "Court of Appeal Parts Company with the EPO on software patents", (2007) 23 Computer L & Sec R at 199.

Ballardini, Rosa Maria, ''Software patents in Europe: the technical requirement dilemma", (2008), 3 Journal Intell Prop L & Prac 9 563.

Basheer, Shamnad, & Wilkof, Neil eds, *Overlapping Intellectual Property Rights*, (Oxford, U.K :Oxford University Press, 2012)

Bell, Gordon, "Stars: Rise and Fall of Minicomputers" IEEE Xplore (17 March 2017), online: Engineering and Technology History Wiki http://ethw.org/Rise_and_Fall_of_Minicomputers >.

Bender, David, "Trade Secret Protection of Software", (1969-1970) 38 Geo. Wash. L. Rev. 909.

Berkowitz, Peggy, “Canada Is Drafting New Copyright Law to Satisfy Grievances of U.S. Concerns”, *Wall Street Journal* (29 April 1986) online: *Wall Street Journal* <http://search.proquest.com/docview/398055666?rfr_id=info%3Axri%2Fsid%3Aprimo>.

Bessen, James and Hunt M., Robert**, “**An Empirical Look at Software Patents”, online (2007) 16:1 Journal of Economics & Management Strategy at 158 < http://onlinelibrary.wiley.com/doi/10.1111/j.1530-9134.2007.00136.x/epdf>.

Beth Gaze, *Copyright Protection of Software* (Sydney, Australia: The Federation Press, 1989).

Bill O, and Mike E., John K. Wayne O. **“***Introduction to the New Mainframe: z/OS Basics”*, (02 January 2012), online IBM Readbooks < http://www.redbooks.ibm.com/abstracts/sg246366.html?Open> .

Biermann, Alan W., *Automatic Programming: A Tutorial on Formal Methodologies*, (London: Academic Press Inc., 1985.

Boyle, Carolyn, Nicholas Fox, & Sian O’Neill eds, *Intellectual property in Electronics and Software: A Global Guide to Rights and Their Applications (London: Globe business Publishing Ltd. 2013).*

Braithwaite, William J., “Derivative Works in Canadian Copyright Law”, (1982) 20: 2 Osgoode Hall LJ 192.

Bray, Robert, *The European Union "Software Patents" Directive: What is it? Why is it? Where are we now?,* (2005) Duke L & Tech Rev11 1.

Canadian Intellectual Property Office - Manual of Patent Office Practice”, March 2007 at c.12 and c.16.

Canadian Intellectual Property Office, Examination Practice Respecting Computer-Implemented Inventions, PN 2013-03.

Cheung, Cheryl, “A Leading Canadian IP Case: Copyright for Computer Software” **Deeth** Williams Wall **(**13 March 2013), online: Deeth Williams Wall < http://www.dww.com/articles/a-leading-canadian-ip-case-copyright-for-computer-software**>.**

Chingale, Ravindra, ‘‘Alice and software patents: implications for India”, (2015), 10 J Intell Prop L & Prac. 5.

Clark, Jane E., & George E. Fisk, “Hardware and Software Protection in Canada” online: (1990) X 10 Computer L.J. <http://repository.jmls.edu/cgi/viewcontent.cgi?article=1424&context=jitpl>.

Conrad Delbert, Seaman, "Contextualizing the Software Patent Debate in Canada: A Practical Approach to Policy Development”, (2014) 3:1 97 Osgoode Hall Review of Law and Policy 3.1 97 at 103.

Cook, Trevor, ed., *Sterling on World Copyright Law*, 4th ed (London, UK: Sweet & Maxwell, 2015).

Cornish, William, Pamela Samuelson, & Thomas Vinje , “Does Copyright Protection under the EU Software Directive Extend to Computer Program Behavior, Languages and Interfaces?” online: (2012) European Intellectual Property Review <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1974890>.

Craig, Carys, Barry Sookman, Steven Mason, and, *Copyright Cases and commentary on the Canadian and International Law*, 2nd ed., (Toronto, Canada: Carswell, 2013

Craig, Carys J, “The Evolution of Originality in Canadian Copyright Law: Authorship, Reward and the Public Interest”, (2005) Osgoode Hall LJ

Crtsinger, Cathy E., “Patent: Patentability: Computer Software,: AT&T Corp. v. Excel Communications, Inc.”, (2000) 15: 1 Berkeley Tech LJ, at 166.

Chung, Haewon, “Lessons from Bilski”, online: (2011) 9 CJLT 1 179 < https://ojs.library.dal.ca/CJLT/article/view/4846%3E>.

Digital Millennium Copyright Act, online: The Electronic Frontier Foundation <https://www.eff.org/issues/dmca>.

Diver, Laurence, “Would the current ambiguities within the legal protection of software be solved by the creation of a sui generis property right for computer programs”, online: (2008) 3 J Intell Prop L & Practice 2 < http://jiplp.oxfordjournals.org/content/3/2/125.abstract> .

Dixit, Sangeeta and Dixit, J. B. , , Fundamentals of Computer Programming and Information Technology, (India: Laxmi Publications, 2005).

Downing, Douglas, Michael Covington, Melody Covington, and Catherine Anne Covington, *Barron's Dictionary of Computer & Internet Terms, 10th ed.*, (Barron's Educational Series: 2009).

England, Paul, ‘‘Computer-related inventions: from CFPH to Macrossan”, (2007), 2 J Intell Prop L & Prac. 5 305.

European Commission, “Digital Economy and Society: The Internet of Things”, http://ec.europa.eu/digital-agenda/en/internet-things>.

Evetkovski, Trajce, *Copyright and Popular media: Liberal Villains and Technological Change*, (London, UK: Palgrave Macmillan, 2013).

Farrar, Anne Layne- & Evans , David S., “Software Patents and Open Source: The Battle over Intellectual Property Rights, (2004) 9:10 Va JL & TECH.

Friedman, Mark M. Friedman, “Copyrighting Machine Language Computer Software-The Case Against”, online: (1989) 9 Computer L.J. 1 http://repository.jmls.edu/cgi/viewcontent.cgi?article=1430&context=jitpl .

Galler, Bernard A., *Software and Intellectual protection: Copyright and patent Issues for Computer and legal Professionals*, (London, UK: Quorum Books, 1996).Gibby, John A., “Software Patent Developments: A Programmer's Perspective”, (1997) 23 Rutgers Computer & Tech LJ 293

González, Andrés Guadamuz, Software Patentability: Emerging Legal Issues, IP and Software (06 December 2008), online: WIPO Magazine <http://www.wipo.int/wipo_magazine/en/2008/06/article_0006.html>

Gonzalez, Andres Guadamuz, ‘‘The software patent debate’’, (2006) 1 Journal Intell Prop L & Pract 3.

Gorman, Robert A., “Comments on A Manifesto Concerning the Legal Protection of Computer Programs,” (1994-1996) 5 Alb. L.J. Sci. & Tech. 277.

Gratton, Eloise, “Should Patent protection be Considered for Computer Software- related Innovations”, (2003) VII Computer L Rev & TJ.

Gervais, Daniel, “The Derivative Right, or Why Copyright Law Protects Foxes Better than Hedgehogs”, (2013) 15: 4 Vand J Ent L & Prac 785.

Grigoriadis, Lazaros G., “Exhaustion and Software Resale Rights in Light of Recent EU Case Law”, online: 2014) 5 J. Int’l Media & Entertainment Law 1 <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2403554>. Guiho G., and Biermann, Alan, eds, *Computer Program Synthesis Methodologies: Proceedings of the NATO Advanced Study Institute*, (Bonas, France: Springer, 1982).

Gurry, Francis, Frederick M. Abbot, & Thomas Cottier, *International Intellectual Property in an Integrated World of Economy*, (New York: Wolters Kluwer, 2015) 692

Hancock, Kimbery, “1997 Canadian Copyright Act Revisions”, (1998) 13 Berkeley Tech. L.J. <*http://scholarship.law.berkeley.edu/btlj/vol13/iss1/33/* >.

Handa, Sunny, *Copyright in Canada*, (Markham, Ontario: Butterworths Canada Ltd., 2002).

Handa, Sunny, “Reverse Engineering Computer Programs under Canadian Copyright Law” (1994) 40 McGill LJ 621.

Hatch, Orrin G., “Better Late Than Never: Implementation of the 1886 Berne Convention”, (1989), 22:2 Cornell Int’l LJ, 1 169.

Hayes, David, ‘‘Brief History of Software: From main frame to mobile’’, Software IP: The 20th Annual BCLT/BTLJ Symposium – Intellectual Property Protections for Computer Programs Past, Present, and Future delivered at The 20th Annual BCLT/BTLJ Symposium of **U.S,** UC Berkeley School of Law**,** April 14th, 2016) [unpublished].

Hettinger, Edwin C, “Justifying Intellectual Property”, online: (1989) 18 Philosophy & Public Affairs 1 at 41, 49 https://www.jstor.org/stable/pdf/2265190.pdf

Hollaar, Lee A., *Legal Protection of Digital Information*, (Washington DC, USA: BNA Books, 2002).

Hormby, Tom, “VisiCalc and the Rise of the Apple II” Apple History (25 September 2006), online: Low End Mac’s Online Groups < http://lowendmac.com/2006/visicalc-and-the-rise-of-the-apple-ii/>.

Hugenholtz , P. Bernt, ed, *The Future of Copyright in a Digital Environment*, (The Hague: Kluwer Law International, 1996).

Ignatin, Gary R., "Let the Hackers Hack: Allowing the Reverse Engineering of Copyrighted Computer Programs to Achieve Compatibility" (1992) 140 U. Penn. L. Rev. 1999.

Information Society Technologies Advisory Group, “The Missing Key Enabling Technology Toward a Strategic Agenda for Software Technologies in Europe” , online: (2012) EU Commission < https://ec.europa.eu/digital-single-market/en/news/software-technologies-missing-key-enabling-technologies-istag-working-group-software>.

James E, Bessen , “A Generation of Software Patents”, online: (2011), Boston Univ. School of Law, Law and Economics Research Paper No 11-31 & Berkman Center Research Publication No. 2011-04, <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1868979 >.

Johns, Adrian, *Piracy the intellectual property wars from Gutenberg to Gates*, (Chicago: The University of Chicago Press, 2009).

Johnson, Luanne, “*Creating the Software Industry Recollections of Software Company Founders of the 1960s”*, IEEE Annals of the History of Computing, (07 August 2002), IEEE Xplore Digital Library at 14< http://ieeexplore.ieee.org/document/988576/>.

Kief F., Scott and James E Daily and, *Perspectives on Patentable Subject Matter*, (New York, U.S.A.: Cambridge University Press, 2015.

Kittredget, C. Mark, “The Federal Circuit and Non-patentable Subject Matter Under In Re Alappat and in Re Warmerdam”, (1995) 11 Santa Clara computer & High Tech. L.J. 261.

Koepsell, David, *Innovation and Nanotechnology: Converging Technologies and the End of Intellectual Property*, (New York, U.S.: Bloomsbury Academic, 2011).

Krauthaus, Patricia Ann & Nimmer, Raymond T., "Software Copyright: Sliding Scales and Abstracted Expression", (1995) 32 Hous. L. Rev. 317.

Kramer, Karen J., "Extending Copyright Protection to a Computer Program's Structure. Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc. 797 F.2d 1222 (3d Cir. 1986)", online: (1987) 65:2 Wash. U. L. Q. 471 < http://openscholarship.wustl.edu/law_lawreview/vol65/iss2/6/>.

Kremer, Michael, Patent buy outs: A mechanism for Encouraging Innovation 113Q.J.Econ.1137(1998).

Kretschmer, Martin, "Software as Text and Machine: The Legal Capture of Digital Innovation", online: (2003) JILT, < https://www2.warwick.ac.uk/fac/soc/law/elj/jilt/2003_1/kretschmer/

Leith, Philip, *Software and Patents in Europe*, (Cambridge, UK: Cambridge University Press, 2007).

Lessig, Lawrence, "Law Regulating Code Regulating Law", (2003) 35 Loy. U. Chi. L. J 1.

Lessig, Lawrence, *Code: And Other Laws of Cyberspace*, (New York, U.S.A: Basic Books, 2006).

Lessig, Lawrence, "The Law of the Horse: What Cyber law Might Teach", (1999) 113 *Harv L Rev* 501.

Lessig, Lawrence & Stallman, Richard, Free Software, Free Society: Selected Essays of Richard M. Stallman., 2nd ed. (Boston: Createspace, 2009).

Liverzani, Amanda, "Fate of Software Patents Still Unclear Following SCOTUS Decision in Alice v. CLS Bank", Harv JL & Tech (28 June 2014), online: Harvard Journal of Law & Technology Digest <*http://jolt.law.harvard.edu/digest/fate-of-software-patents-still-unclear-following-scotus-decision-in-alice-v-cls-bank*>

Locke, John, *Second Treatise of Government*, Book II, Ch. V, 1690, at para 26 & 27 <http://www.earlymoderntexts.com/assets/pdfs/locke1689a.pdf >.

McKeough J., ‘‘Apple Computer Inc. V. Computer Edge Pty Ltd”, A Case Note’’, (1984) UNSWLJ 162.

McCormack, Stuart C., ed., *Intellectual Property Law of Canada 2nd ed (New York, U.S.A: Juris Publishing, Inc., 2010).*

McKeough J., ‘‘Apple Computer Inc. V. Computer Edge Pty Ltd”, A Case Note’’, (1984) UNSWLJ 162.

Macmillan, Kathleen Gilbert-, “Intellectual Property Law for Reverse Engineering Computer Programs in the European Community”, (1993) 9 Santa Clara High Tech. L.J. 247.

Makarenko, Jay , “Copyright Law in Canada: An Introduction to the Canadian Copyright Act”Mapleleafweb (13 March 2009), Judicial System & Legal Issues < http://www.mapleleafweb.com/features/copyright-law-canada-introduction-canadian-copyright-act.html>.

Mann, Ronald J., “Commercializing Open Source Software: Do Property Rights Still Matter?” (2006) 20: 1 Harv JL & Tech.

Mann, Ronald J., John R. Allison, & Abe Dunn, “Software Patents, Incumbents, and Entry”, (2006-2007) 85 Tex. L. Rev. 1579.

McKenna, Barrie, “Canada needs tougher drug patent protection: Report” The Globe and Mail (23 August 2012) online: The Globe and Mail < http://www.theglobeandmail.com/report-on-business/canada-needs-tougher-drug-patent-protection-report/article562405/>.

Mehta, Arun, “The Absurdity of Software Patents”, (11 December 2003) *http://world-information.org/wio/readme/992006691/1078487756*

Menell, Peter S. , “Envisioning Copyright Law's Digital Future”, Online: (2002-2003) 46 New York Law Review < https://papers.ssrn.com/sol3/papers.cfm?abstract_id=328561>.

Menell, Peter S., “Tailoring Legal Protection For Computer Software”, (1987), 39 Stan L Rev 6.

Mihm, Mickey T., “Software Piracy and the Personal Computer: Is the1980 Software Copyright Act Effective?”, (1983) 4 Computer L.J. 1

Miles, Matthew B., and Weitzman, Eben, Computer Programs for Qualitative Data Analysis: A Software Sourcebook

Miyashita, Yoshiyuki, ‘‘International protection of Computer software’’, (1991), 11 Computer L.J. 41<http://repository.jmls.edu/cgi/viewcontent.cgi?article=1390&context=jitpl>.

Morris, Emily Michiko, “What Is “Technology”?”, online: (2014) B.U. SCI. & TECH. L. (2014) http://fstp-expert-system.typepad.com/files/92-e.-morris_what-is-technology_iu_i.n..pdf .

Nolet , Dominique, “The Protection of Icons and Interfaces by Industrial Design” *ROBIC* online: < http://newsletter.robic.ca/nouvelle.aspx?lg=EN&id=241>

Ogilvie, John W.L., ‘‘Defining Computer Program Parts under Learned Hand's Abstractions Test in Software Copyright Infringement Cases’’, Note, (1993) 91 Mich. L. Rev 526

Osenga, Kristen, “Debugging Software’s Schemas”, (2014) 82:6 Geo Wash L Rev 1833.

The Harvard Law Review Association, “Patent Law - Patentable Subject Matter - Federal Circuit Applies New Factors in Deciding Patentability of a Computer Program. - Ultramercial, LLC v. Hulu, LLC’, 657 F.3d 1323 (Fed. Cir. 2011), reh'g and reh'g en banc denied, No. 2010-1544, 2011 U.S. App. LEXIS 25055 (Fed. Cir. Nov. 18, 2011),” online: (2012) 125 Harv. L. Rev. 2167 at 2169 <http://www.jstor.org/stable/23214434?seq=1#page_scan_tab_contentsPerry, Lawrence and Hugh Brett, *The legal Protection of Computer Software*, (Oxford, UK: ESC Publishing Ltd, 1981).

Phillipson, Graeme, “A Short History of Computer”, (2004).

Phillips, Jeremy, “Save Analytical Software”? That’s not what SAS stands for…” The IPKat blog (January 2013) The IPKat blog, online: The IPKat: intellectual property news and fun for everyone < http://ipkitten.blogspot.ca/2013/01/save-analytical-software-thats-not-what.html>.

Phillips, John C., ‘‘*Sui generis* Intellectual Property Protection for Computer Software’’, (1992) 60 Geo. Wash. L. Rev. 997.

Quinn, Gene, “The history of software patents in the United States” IPWatchdog (03 October 2014), online: Patent bar Review <http://www.ipwatchdog.com/2014/11/30/the-history-of-software-patents-in-the-united-states/id=52256/>.

Reger, Christina M., “Let's Swap Copyright for Code: The Computer Software Disclosure Dichotomy”, (2004) 24 Loy LA Ent LR 215

Reidenberg,Joel R., “*Lex Informatica*: The Formulation of Information Policy Rules through Technology”, online: (1998) 76 Tex. L. Rev. 3< http://ir.lawnet.fordham.edu/faculty_scholarship/42/>.

Report of an Industry Expert Group on a European Software Strategy, Playing To Win In the New Software Market: Software 2.0: Winning For Europe, (June 2009 Version 3.5).

Reynolds, Graham, “*Towards a Right to Engage in the Fair Transformative Use of Copyright-Protected Expression”, in Michael Geist, ed, From “Radical Extremism” to “Balanced Copyright”: Canadian Copyright and the Digital Agenda* (Toronto: Irwin Books, 2010).

Risch, Michael, “Hidden in Plain Sight” , Online: (2016) Villanova Public Law and Legal Theory Working Paper Series, < https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2761100##> .

Roberts, Tom, Intellectual and Industrial Property I: Introduction to Patents, Lecture Notes, (College of Law, University of Saskatchewan, 2015).

Robertson, Ronald, *Legal protection of Computer Software*, (London, UK: Longman law, 1990).

Rognstad, Ole-Andreas, “Legally Flawed but Politically Sound? Digital Exhaustion of Copyright in Europe after UsedSoft” online: (2014) 1 Oslo L Rev < <https://www.journals.uio.no/index.php/oslawreview/article/view/977 >.

Samuelson, Pamela, “CONTU Revisited: The Case against Copyright Protection for Computer Programs in Machine readable Form”, (1984) Duke LJ 663.

Samuelson, Pamela, ‘‘The Uneasy Case for Software Copyrights Revisited”, (2011) 79 Geo. Wash. L. Rev. 1746

Samuelson, Pamela, Randall D., Mitchel D., J.D. Reichman, “A Manifesto Concerning the Legal Protection of Computer Programs”, (1994) 94 *Colum L Rev* 2308-2431

Samuelson, Pamela, ‘‘*The U.S. Digital Agenda at WIPO*, 37 *Va. J. Int'l L.* 369 (1996)’’ < http://scholarship.law.berkeley.edu/facpubs/882

Samuelson, Pamela, “Comparing U.S. and EC Copyright Protection for Computer Programs: Are They More Different Than They Seem?” (1993)13 J.L. & Com. 279.

Samuelson, Pamela, “Reflections on the State of American Software Copyright Law and the Perils of Teaching It”, online: (1988) 13 Colum.-VLA J.L. & Arts 61 (1988) < http://scholarship.law.berkeley.edu/facpubs/128/>.

Samuelson, Pamela, “Modifying Copyrighted Software: Adjusting Copyright Doctrine to Accommodate a Technology”, online: (1988) 28 Jurimetrics J 179 < http://scholarship.law.berkeley.edu/facpubs/653/>.

Samuelson, Pamela, “Why the look and feel of software user interfaces should not be protected by copyright law”, online :(1989) 32Communications of the ACM 5 <http://www.foo.be/andria/docs/p563-samuelson.pdf>.

Saylor Foundation, “Brief History of Computer Systems, Software, and Programing”, <http://www.saylor.org/site/wp-content/uploads/2014/07/CS101-1.1-Brief-History-of-Computer-Systems-Software-and-Programming.pdf.

Scmitz, Sandra V.I., The Struggle in Online Copyright Enforcement: Problems and Prospects (Luxemburg: Hart Publishing, 2015).

Scotchmer, Suzanne and Samuelson, Pamela, “The Law and Economics of Reverse Engineering”, (2001) 111 Yale L.J. 1575.

Sherman, Brad & Bently, Lionel , *Intellectual Property Law*, 3rd ed (New York, U.S.A: Oxford University Press, 2009).

Sobin, Sturgis M. and Lande, Robert H., “Reverse Engineering of Computer Software and U.S. Anti-trust Law”, (1996) 9 :2 Harv JL & Tech.

Steinbrenner, Stefan, “The patentability of computer-implemented inventions”, EPO (24 March 2011) <http://archive.is/e-courses.epo.org>.

Stokes, Simon, *Art and Copyright*, Oxford (Oxford, UK: Hart Publishing, 2012).

Story, Alan, "Intellectual Property and Computer Software: A Battle of Competing Use and Access Visions for Countries of the South", (ICTSD and UNCTAD, 2004)

Swinson, John, "Copyright or Patent or Both: An Algorithmic Approach to Computer Software Protection," (1991) 5 Harv JL & Tech 146.

Szabo, Howard K., "International Protection of Computer Software: The Need for Sui Generis Legislation", (1986) 8 Loy L.A. Int'l & Comp. L. Rev511.

Thomas, Robert E., "Debugging Software Patents: Increasing Innovation and Reducing Uncertainty in the Judicial Reform of Software Patent Law", (2008) 25 Santa Clara Computer & High Tech. L.J.

Toeniskoetter, Steven B., "Protection of Software Intellectual Property in Europe: An Alternative Sui Generis Approach", online: (2007) 10 INTELL. PROP. L. BULL 65 <http://heinonline.org/HOL/Page?handle=hein.journals/iprop10&div=9&g_sent=1&collection=journals>.

Tusssey, Deborah, *Complex Copyright: Mapping the Information Ecosystem*, (England: Routledge, 2012).

Tutorials Point (I) Pvt. Ltd, "Computer Programing Tutorial", Tutorials Point (2014) online: Simply Easy learning <https://www.tutorialspoint.com/computer_programming/computer_programming_pdf_version.htm >.

The Concise Oxford Dictionary (11th ed., 2004).

U.S. Gov't Accountability Office, GAO-13-465, Intellectual Property: Assessing Factors That Affect Patent Infringement Litigation Could Help Improve Patent Quality 12 Fig.1 & N.27 (2013).

U.S. Patent No.7,346,545

Vaver, David, *Essentials of Canadian Law: Intellectual Property Law: Copyright, Patents, Trademarks* (Concorde Ontario: Irwin Law Concorde Ontario, 1997

Weichselbaum, Mindy J., "The EEC Directive on the Legal Protection of Computer Programs and U.S. Copyright Law: Should Copyright Law Permit Reverse Engineering of Computer Programs?" (1997) 3 Buffalo Journal of International Law 519.

Westermann, Hannes, How to treat software in the intellectual property framework (LLM thesis, Lund University Faculty of Law, 2016) [unpublished].

WIPO, Guide to the copyright and related rights treaties administered by WIPO and glossary of copyright and related rights terms, (2003).

WIPO Intellectual Property Handbook, "Technological and Legal Developments in Intellectual Property", (2nd ed., WIPO PUBLICATION No. 489 (E): 2004).

Wiseman, Leanne and Sherman, Brad, ed, *Copyright and the Challenge of the New*, (The Netherlands: Kluwer Law International, 2012).

Working group on Libre Software, ''Free Software / Open Source: Information Society Opportunities for Europe?'', EU commission Community Research and Development Information Center (23 February 2000), Online: EU Commission News & Events < http://cordis.europa.eu/news/rcn/14374_en.html>.

Vaver, David, Copyright Law: Recent Canadian Developments", Online: (1988) 16 *Australian Business Law Review* <http://search.proquest.com/docview/223515078?OpenUrlRefId=info:xri/sid:primo&accountid=14739

Yao Dennis, Anton, James J., and Hillary Greene, "Policy Implications of Weak Patent Rights", (2006) 6 Harvard Business school Innovation Policy and the Economy

Yeh, Brian T., "An Overview of the "Patent Trolls" Debate, Prepared for Members and Committees of Congress", CRS Report for Congress

(16 April 2013) online: < https://archive.org/details/R42668AnOverviewofthePatentTrollsDebate-crs>

Ypersel, Tanguy V and Shavell , Steven, "Rewards Versus Intellectual Property Rights", (2001) 44 J.L & Eco.525.

Zhu, Feng Josh and Lerner, “What is the impact of software patent shifts? Evidence from Lotus v. Borland”, online: (2007) Int. J. Ind. Organ. 25 < http://www.nber.org/papers/w11168> .

Zittrain, Jonathan, “Normative Principles for Evaluating Free and Proprietary Software”, (2004) 71 U *Chicago L Rev.*