CENTRAL EUROPEAN UNIVERSITY

Legal Studies Department

SOFTWARE PROTECTION BETWEEN PATENTS AND COPYRIGHT: A COMPARATIVE ANALYSIS

by Anna Mychka

In partial fulfillment of the requirements for the degree of Master of Laws

LL.M. SHORT THESIS

COURSE: International and Comparative Intellectual Property Law

PROFESSOR: Caterina Sganga

Central European University

1051 Budapest, Nador utca 9

Hungary

EXECUTIVE SUMMARY

The purpose of this paper is to identify whether patent or copyright protection is better for software programs. Software industry has been intensively developing over the past few decades. Its novelty, dynamics and commercial application has attracted numerous and versatile businesses while complexity and multiplicity fascinated academia. Somewhere in between stands law. It has to walk the tightrope, balancing between interests of developers, businesses and public. At present, patentability of software is more inherent to the U.S. while Europe hesitates if it should remain with customary copyright or switch to patentability. The legal uncertainty has led to the situation when software enjoys protection from both copyright and patent. The most effective way to choose a proper means is to assess which is more beneficial to all interested parties. The proper solution lies in the realm of economic analysis of law. It offers an insight into practical results of legal provisions and how they affect economy and society. There is no proof that traditional incentive theory behind patentability works for software sector. On the other hand, copyright limitation to “written” elements of software is also ill-equipped to combat unfair use. Software is unconventional matter therefore it requires unconventional solutions. It is suggested that legislators should abandon traditional copyright-patent conflict and instead produce a unique however balanced provisions to specifically meet the needs of this modern phenomenon.

Table of Contents

Introduction

Throughout the last decade, a heated debate over the means of legal protection for software has been raging in Europe due to the dual form available for computer-implemented inventions. An extensive umbrella protection may be granted both by patent and copyright. A software developer may opt to one of them or undertake both, contemplating financial matters, time costs and the nature of protection she deems most effective. Since there is no harmonized law in Europe, which is the important player on the global market of software products, the diversity of regulations, their interpretations by courts and European Patent Office (EPO) cause major uncertainties in the software industry.[1]

The protection regimes for intellectual property rights - copyright, patent and trade secrets - offer different scope and nature of protection and, in order to qualify for any, a "work" should meet certain requirements set out in relevant legal provisions. Copyright protection is granted automatically at the moment of creation of literary or artistic work, therefore giving author rights exclusive right to reproduce, adapt, distribute, perform, and display the work. Patent, on the other hand, is government-granted legal right to exclude others from making, using or selling the patented invention for a certain period of time.[2] Therefore, copyright protects 'original expression' – in the case of computer software: the originally coded program – against direct copying. "Patents protect

[1] Existence of software industry as such is disputable; it is more a diverse industry involving individuals and small firms as well as multinational companies. They offer ether separate software programs or goods that include elements of software or both.

[2] Invention - A patentable device or process created through independent effort and characterized by an extraordinary degree of skill or ingenuity BLACK'S LAW DICTIONARY (West 9th ed) (2009)

inventive ideas – in the case of a software related invention: the exclusive right to apply the idea."[3]

European countries regulate intellectual property issues by following the provisions of the European Patent Treaty (EPT), TRIPS agreement, Berne and Paris conventions. Since becoming commercially attractive and widely used in the middle 80's, software has been difficult to classify in order to assign it to a certain category of intellectual property protection by virtue of its binary nature – it is debated if it represents a variation of mathematical algorithm or invention. Commentators have sought to classify it under copyright, patents, both, trade secrets or even as *sui generis* software right.[4] Finally, emphasizing the written form of "algorithm" of source code, software programs have been classified as literary work, hereby eligible for copyright protection under Berne Convention, TRIPS agreement.

However, in due course of EPO practice, the Office has allowed certain degree of patentability of software programmes. Despite clear provisions of the above agreements and treaties, practice and case law in Europe have allowed the limited patentability of so-called "computer-implemented inventions" that involve a technical effect (or contribution or process).[5] The term was coined in the judicial practice and basically relies on the principle that even through software program is not patentable as such, if it

[3] "THE PATENTABILITY OF COMPUTER PROGRAMMES. DISCUSSION OF EUROPEAN LEVEL LEGISLATION IN THE FIELD OF PATENTS FOR SOFTWARE, EUROPEAN PARLIAMENT.PDF 5, http://www.europarl.europa.eu/meetdocs/committees/juri/20020619/SoftwarePatent.pub.pdf (last visited Mar 27, 2012)

[4] Andre´s Guadamuz Gonza´lez, *The software patent debate*, JNL. INTELLECTUAL PROPERTY LAW AND PRACTICE 1–11, 2 (2006), http://www.wipo.int/edocs/mdocs/copyright/en/wipo_ip_cm_07/wipo_ip_cm_07_www_82574.pdf

5 *"VICOM"* (1987) 2 EPOR 74; *Merrill Lynch's Application* (1989) RPCC 561; *Gale* (1991) RPC 305

provides for certain "tangible" effect, technical advance or industrial applicability it may be patentable. Since, practically all, software programs demonstrate, to some extend, "tangible" effect, such wording renders most of programs patentable. If patent application is drafted "smartly" and able to convince patent reviewer that there is such effect, nearly all programs may get patent. Therefore, it contributes to the "[I]mpression that the way a patent claim is worded is often decisive" in the outcome of patent claim. [6] It is notoriously complicated to estimate the number of issued patents for software as EPO does not have a separate category in its system, however some statistical data claims that until 2005 the EPO issued from 20,000 to 30,000 patents for software programs.[7] At certain point, the European Commission considered to formally recognizing patentability of software in EU. A number of reports have been delivered to European Commission and Parliament to clarify how efficient would patent address software industry needs, most notably, "*The patentability of computer programmes. Discussion of European level legislation in the field of patents for software"*[8], and "*The economic impact of Patentability of Computer programs*".[9] The major source of "inspiration" for software patentability came to EU from the United States. Consequently, the above reports, as well as numerous academic opinions originating from distinguished scholars in law and economy, concentrated on US to study the effect

[6] Ibid 2 at 8

[7] EUROPEAN SOFTWAR PATENT STATISTICS FOUNDATION FOR FREE INFORMATION INFRASTRUCTURE (FFII), http://eupat.ffii.org/patents/stats/index.en.html (last visited Mar 29, 2012)

[8] THE PATENTABILITY OF COMPUTER PROGRAMMES. DISCUSSION OF EUROPEAN -LEVEL LEGISLATION IN THE FIELD OF PATENTS FOR SOFTWARE, http://www.europarl.europa.eu/meetdocs/committees/juri/20020619/SoftwarePatent.pub.pdf

[9] ROBERT HART ET AL., THE ECONOMIC IMPACT OF PATENTABILITY OF COMPUTER PROGRAMS. REPORT TO THE EUROPEAN COMMISSION. (Intellectual Property Institute), http://ec.europa.eu/internal_market/indprop/docs/comp/study_en.pdf

of patents in software industry. They posed the question whether US approach could be adapted for EU. The conclusions of reports as well as scholars are inconsistent and in practice do not provide the answer whether patentability of software would have better effect on software industry than copyright protection.

It should be pointed out that in the United States, copyright and patent are also two major legal tools used to protect intellectual property rights of software developers.[10] Nevertheless, in the course of time, U.S. turned into more patent-friendly jurisdiction for software through judicial interpretation and precedents principle of common law jurisdiction. Today it is estimated that the US Patent Office issues about 20,000 software patents annually.[11] Supporters of patentability for software argue that the leading position of the US in software industry evolved because it had software patent at its disposal and this greatly promoted and enhanced research and development investment into the field. However it is troubling that copyright protection is simultaneously available in the US and allows software creator to potentially monopolize an entire field which its software covers.[12]

In view of EPO practice and in attempt to harmonize law European Commission offered to enact EU Directive with regards to patentability of software but it turned out to be major failure. In 2005, the European Parliament voted by a large majority to reject the Directive on Patentability of Computer-Implemented Inventions. It was also rejected by both supporters and opponents of patentability of software, thus fuelling further debate

[10] Andrew Nieh, *Software Wars: The Patent Menace*, 55 NEW YORK LAW SCHOOL LAW REVIEW 296

[11] James E. Bessen & Robert M. Hunt, *An Empirical Look at Software Patents*, SSRN ELIBRARY , 3 (2004), http://papers.ssrn.com/sol3/papers.cfm?abstract_id=461701&rec=1&srcabs=886905 (last visited Mar 29, 2012)

12 *Feist Publ'ns, Inc. v. Rural Tel.Ser. Co*., 499 U.S. 340 (1991)

over the issue. The stumbling block where participants to discussion of Directive failed to agree was the definition of what software is and what is computer-implemented invention that involves technical effect as the Directive offered to allow patents to the latter but remain software *per se* under copyright protection. Article 5 of proposed Directive defined computer-implemented inventory in such a fashion that would potentially allow to patent software as such without any technical or tangible effect. Moreover, it was argued that all computer programs, in that way or another, have some technical effect as they are run on hardware that produces tangible results. The debate over this issue mainly led to Directive's downfall. However, most commentators suggest that the battle is not over and soon there will be another attempt to agree on patentability of software.

The realm of patentability prospects for software in EU is viewed as contemporary and urgent. While big international companies support patentability of software, small and medium businesses coupled with academics, scholars and independent developers are strongly against it. Software has practically infiltrated every aspect of modern life. We hardly find today everyday things like the phone or the stove that run without at least trivial elements of software. It is vital to establish which legal vehicle is more suitable to provide clarity and guidelines for those involved into the field, safeguard incentive for software developers, ensure innovation and progress and benefit society in the long run.

This thesis seeks to identify which approach is more reasonable and meets requirements of economic and social beneficial effects of software development for

programmers, businesses and public. Comparative analysis of U.S. and European approaches in this thesis is focused on the consequences of protection granted by a particular legal vehicle.

Chapter I review historical development of legal framework in the field to ensure basic understanding of the nature of conflict and inconsistency existing today. Also, it provides the general overview of the structural elements of software as well as its classification. Chapter II discusses patents, eligibility of software components to patentability requirements, scope of protection under this legal tool and its impact from economic standpoint. Chapter III examines copyright and its suitability to accommodate needs for software protection, extend of protection and impact in the industry. In addition, it deliberates on the issue of open source software. Finally, Chapter IV explores the area of economic efficiency of various legal modes available for software and evaluates their performance.

The conclusions from this thesis identify that both copyright and patents are disputable as none offers proper balance between private incentive and public good. The proper solution for EU dilemma is to tailor provisions specifically for software taking into account economic efficiency and impact, support of innovative incentive and maintenance of public interest as well as lifespan of software products.

Chapter I - Software within legal framework.

Behind legal concept there should be an understanding of what is regulated and why it should be regulated in a certain way. In the following subsections I briefly explain the timeline of software development and applicable legislation as well as generally characterize types of software.

I.1Brief historical overview.

For the sake full comprehension of the topic, it is useful to devote some attention to the history of software evolution with focus on regulations applied to it. As it was pointed out earlier, the thesis is concentrated on US and EU jurisdictions. It is not unusual, in view of the fact that majority of cutting edge programs developed for the past few decades, accrue to them. Therefore concise insight encompasses only EU and the U.S.

I.1.1Development in the U.S.

As it was explained by Professor M. Lemley, during the early stages of computer industry most computer software was provided by computer manufacturers along with the hardware. There was little or no interest in protecting software technology separately because patent protection for computer hardware adequately rewarded innovation.[13] With computer evolution, appeared specialty software firms to offer customized and/or general purpose software thus coming into competition with hardware manufacturers. However, the industry targeted mainly specialized customers which needed customized solutions to tackle specific problem. Software firms were able

[13] SOFTWARE AND INTERNET LAW 33 (Aspen Publishers 3rd ed) (2006)

to draft contracts for individual customers and exercise control over their proper execution.

Finally, it became profitable for software companies to offer systems and particular application programs to a wider market.[14] At the point when self-sufficient software mass market matured, it was evident that existing legal framework was not able to embrace the needs of the industry.

The Copyright Act of 1976 was introduced to address the issue of new emerging technologies in the U.S. The National Commission On New Technological Uses of Copyrighted Works (CONTU), created by Congress, suggested in its final report that computer programs should be qualified for copyright protection. Following recommendations, Congress approved copyright protection for computer programs and enshrined the relevant provisions in Copyright Act. Under the scope of this Act a "'[C]omputer program' is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result" and it is enveloped by the definition of 'literary works'.[15] "It was soon established that both source code and object code, as well as the code for operation systems were all literary works […] and subject to copyright protection as original expressions."[16]

On the other hand, the Patent Act of 1952 defined that patents are only granted to inventions and processes that fall under the statutory subject matter, thus claims can be

[14] *Id.* at 33

[15] COPYRIGHT LAW OF THE UNITED STATES 2,4, http://www.copyright.gov/title17/circ92.pdf

[16] Nieh at 301

patented if they fall under one of four categories: process, machine, manufacture and composition of matter.[17] Though computer programs were not, initially, intended to be protected by the Patent Act, courts have eventually construed that some computer programs may be defined as "process" and therefore, potentially qualified for patentability if they meet requirements set out for patentability of any other invention, that is novelty and usefulness.

The first Court decision on the matter was issued back in 1972 in *Gottschack v. Benson* where the Supreme Court held that the computer program in question, as such, was mathematical algorithm, therefore not patentable.[18]Yet, already in 1981, in *Diamond v. Diehr,* the Supreme Court held that under certain conditions patent could, indeed, be granted to computer programs as the decisive test is whether the invention involves "transformation and reduction of an article into a different state or thing".[19]

Since the Patent Act and the Copyright Act, Congress has not addressed the issue of legal treatment of software programs. Instead, "a number of court and administrative decisions gradually relaxed the subject matter that restricted the patenting of software-related inventions"[20] and thus established patentability for computer programs which results in about 20,000 software patents granted annually in the US. It is curious that, at the same time, the past decade evidenced a true surge of law suits involving software patents battles of Microsoft, Apple, Motorola, HP, Oracle, etc.

[17] *Id*. at 303

[18] *Gottschalk v. Benson*, 409 U.S.63.

[19] *Diamond v. Diehr*, 450 U.S. at 175

[20] Bessen & Hunt at 3

Economists and lawyers launched a nation-wide discussion as to economic efficiency of software patents and their impact in the field. Edith Penrose and Fritz Machlup elaborated on the topic to "[D]ispel the widespread American misconception that patent protection had always been accepted as unquestionable benefit to the society."[21] The authors suggested to 'to muddle through' patents. Patent system has existed for a long time and abolishing it now would be irresponsible.

Despite the fact that initially software was intended to be protected under copyright, court decisions gradually relaxed requirements for software, accepted it as process and therefore eligible for patentability. It was followed by dynamic development of the industry and many assigned the success to affordable patentable patents. Nevertheless, today lawyers, economist, and scholars question patentability "blessing" because of recent increase in litigation and cost of patents.

I.1.2. Development in Europe

The main scope of IP rights in Europe was enshrined in Berne Convention for the Protection of Literary and Artistic Works (Berne Convention) for copyright and Paris Convention for the Protection of Industrial Property (Paris Convention) for patents. Computer programs were assigned to literary works, therefore considered eligible for copyright protection under Berne Convention. Thought Berne Convention offers world–wide copyright regime, due to the fact that it does not explicitly address the legal status of software, nations adopted their own diverse approaches to patentability of software.

[21] Fritz Machlup & Edith Penrose, *The Patent Controversy in the Nineteenth Century*, 10 THE JOURNAL OF ECONOMIC HISTORY 1–29 (1950)

On the other hand, Paris convention did not specifically define what is excluded from patentability, moreover it did not address the issue of duration for patent.

> "By failing to establish such minimum standards, especially with regard to what is actually patentable, the Paris Convention left global patent protection generally, and computer software protection specifically, in a state of uncertainty".[22]

The advance of global Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS Agreement) brought some degree of clarity as to duration of patent. Also, in Art.27 it has defined the scope of patentability "[F]or any inventions, whether products of processes, in all fields of technology, provided that they are new, involve an inventive step and are capable of industrial application".[23] In its copyright section, namely Art.10, TRIPS addresses computer programs with respect to object and source code and granted them protection under Berne Convention. Thus, limiting definition of computer programs to source and object code as "literature" parts that are automatically protected by copyright, TRIPS has enabled software developers to attain patent protection for other software elements based on functional or behavioural aspects of the program.[24] Moreover, despite the fact that European Patent Convention (EPC) in its Art.52 excluded software from patentable inventions, its Art.53(2) was invoked to a series of

[22] Aaron D. Charfoos, *How Far Have We Come, And Where Do We Go From Here: The Status Of Global Computer Software Protection Under The TRIPS Agreement*, NORTHWESTERN JOURNAL OF INTERNATIONAL LAW AND BUSINESS 1–39, 5 (2002), http://www.kirkland.com/siteFiles/kirkexp/publications/2499/Document1/Charfoos_Northwestern%20Univ%20School%20of%20Law.pdf

[23] TRIPS: AGREEMENT ON TRADE-RELATED ASPECTS OF INTELLECTUAL PROPERTY RIGHTS, http://www.wto.org/english/tratop_e/trips_e/t_agm3c_e.htm#5

[24] Charfoos at 13

cases (IBM 1999, Phillips 2000) to argue that software was excluded from patentability *as such* however if software had technical effect it could obtain patent.[25]

According to Bakels, "[C]opyright and patent protection of computer software are complementary regimes. Copyright protects "original expression" […] the originally coded programme against direct copying. Patents protect inventive ideas […] the exclusive right to apply the idea." [26] Thus software enjoys the best of 'two worlds'.

Recent practice of European Patent Office (EPO) shows that software is not excluded from patentability under Art. 52 of EPT if it is defined as "computer-implemented invention" [27].The main criticism of such practice stems from the impression that a claim for computer program patent may be satisfied depending on wording of an application for patent. Lack of extensive prior art information at disposal of patent examiner to reject trivial or obvious patent claims contributes to this impression.

In an attempt to harmonize law in the field of software, European Commission initiated drafting of a Directive with respect to the patentability of computer-implemented inventions. The debate, that followed, demonstrated that there is no clear understanding which legal protection should software be assigned to. Moreover, both opponents and supporters failed even to come up a with single definition of the subject matter of the Directive. A myriad of terms such as computer program, software, computer-

[25] T 1173/97, "IBM 1999" also known as *Computer program product/IBM, T 1194/97, "Phillips 2000" also Data Structure product/PHILIPS*

[26] Bakels at 5

[27] Computer-implemented invention - an invention whose implementation involves the use of a computer, computer network or other programmable apparatus; with features realized wholly or partly by means of a computer program; PATENTS FOR SOFTWARE&, http://www.epo.org/news-issues/issues/computers/software.html

implemented invention etc. circulate the field. Lack of clarity in defining software program, especially under circumstances of global nature of industry, causes general confusion within legal framework assigned to regulate software one hand and benefit to increase of trivial patents and accumulation of patent thickets on the other.

This short description of the situation in the field in major jurisdiction indeed, shows that a need for clear-cut and distinct provisions is pressing, especially in view of the fast advancement of software industry, its global nature and its increasing omnipresence in our every-day life.

I.2.Structural components of software and classification

One of the reasons of legal uncertainty in the software field is that legislators struggle to understand the nature of software programs. In its own way it resembles Chimera – a monster with the body parts from numerous 'animals'. Software contains mathematical algorithms which can be trivial or resemble 'artistic expression', software may control a process for mere cooperation between elements of hardware or operate sophisticated manufacturing process with tangible effect. It would be complicated to treat every program on *ad hoc* basis but it is also obvious that uniformity offered by copyright or patent is similarly insufficient.

To attain some certainty, I define principal elements of software that came into legal view throughout patent-copyright debate. There are multiple suggestions as to software element that should be taken into account while defining which legal protection and for which structural element and/or type of computer program should be taken into account.

Summarized by Prof. Lipton, the "anatomy" of software for legal purposes could be outlined as unity of source code, object code and Generated User Interface (GUI) [Fig.1].

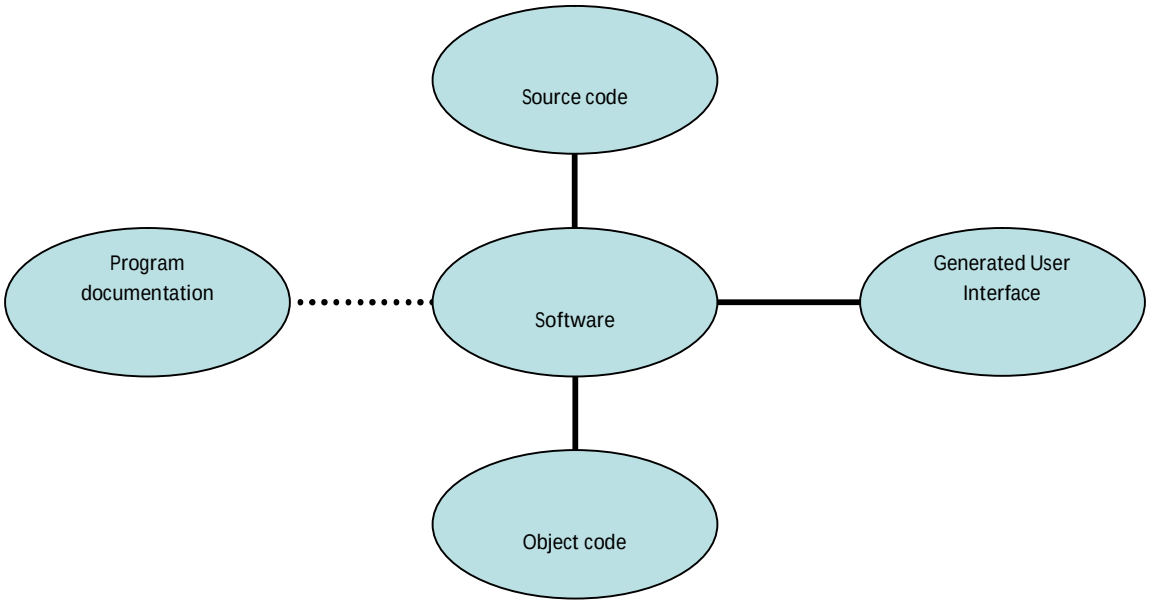


Fig.1

Additionally, though it is not directly incorporated into program structure, program documentation drafted prior to creation of a program should also be taken into account.[28] Source code is a "literary" component and is readable by humans, object code is a binary code readable by machines only and GUI may be interpreted as "look and feel" aspects of program, as well as its structure, sequence and organization.[29]

[28] Jacqueline D. Lipton, *IP's Problem Child: Shifting the Paradigms for Software Protection*, SSRN ELIBRARY , http://papers.ssrn.com/sol3/papers.cfm?abstract_id=901604 (last visited Mar 29, 2012)

[29] Lemley, at 39

In similar vein, I define types of programs. Software is developed to tackle a number of computer-related issues. Thus,

- **Programming software** is used to develop other programs (e.g. debuggers, assemblers);[30]
- **System software** is used to run hardware and ensure cooperation between its units as well as to allow different application programs operate within single platform (e.g. Microsoft Windows, Linux),[31]
- **Application software** is used to carry out various tasks which users want to perform on computers (e.g. word processors, web browsers).[32]

Every type of software may involve trivial parts of source code customary applied to perform certain functions as well as new and inventive code parts, depending on the final objective of the whole program.

In conclusion, Chapter I offered a brief review of the current situation in the field of software protection concentrating on EU and the U.S. jurisdictions as well as provided general structure of software and its classification.

[30] PROGRAMMING SOFTWARE IT DEFINITIONS, http://www.defit.org/2012/02/programming-software.html

[31] SYSTEM SOFTWARE IT DEFINITIONS, http://www.defit.org/2012/01/system-software.html

[32] APPLICATION SOFTWARE IT DEFINITIONS, http://www.defit.org/2012/01/application-software.html

Chapter II - Software protection under patent. Experience and prospects.

In order to provide comprehensive review of the scope of software protection in this chapter, first, I analyze limits of patentability for software. Secondly, in view of the fact that most profound protection under patent umbrella for software, so far, has been provided in the U. S., I study factors that have justified this approach, in particular judicial practice. It is necessary to look into practice of U.S. Patent and Trademark Office (USPTO) with regards to prior art research conducted by it. Thirdly, the reference is made to defensive patents and patent thickets. Finally, having our focus on EU, I look into the practice of European Patent Office and battle against patentability of software in European Union.

II.1.Scope of protection. Background.

Patents are designated to stimulate innovative activity which is assumed to develop competitiveness and economic welfare in general.[33] Governments use patents to reward inventors. Patents are based on utilitarian theory.[34] According to utilitarian principles, if inventors do not receive patent protection for their invention, they will be less likely to contribute to progressive endeavours that ultimately add to the inventory of public knowledge.[35] In other words, should the claim for patent meet specific requirements for patentability, the government grants an inventor a monopoly limited in time for production, sale and use of the invention. Along, an inventor obtains a

[33] Patent is the governmental grant of a right, privilege, or authority. Black's Law Dictionary (9th ed. 2009)

[34] See also incentive theory, the proposition that society grants creators exclusive rights to their intellectual property in order to stimulate further creativity; Black's Law Dictionary (9th ed. 2009)

[35] Nieh at 308

bargaining power to licence his invention for financial gain. In return, inventor should disclose invention and make a contribution to prior art and public domain.[36]

It is argued that if software in the U.S. could not obtain patent, there would be fewer products on the market and eventually software development would come to the halt. Software patents, however purportedly solve this public goods problem, prevent market failure, promote progress and spur innovation.[37]. Patents are limited in time (seventeen-twenty years) and such time is shorter then that of copyright (from fifty to seventy); promote technological progress through early disclosure; give inventor control over his invention (make, sell, use, license and preclude others from doing it without licences), therefore ensure financial gains for invested time and resources, and, potentially, create incentive for further development.

There is empirical evidence that for certain industries, patents are an important factor and explain the willingness to invest in R&D. A number of surveys established importance of patents for U.S. chemical and pharmaceutical industries.[38] But these surveys also show that in many other industries patents are not regarded as either very important or effective in protecting one's innovations.[39]

[36] See also Ordinary Skill in the Art, the level of technical knowledge, experience, and expertise possessed by a typical engineer, scientist, designer, etc. in a technology that is relevant to an invention; Black's Law Dictionary (9th ed. 2009)

[37]Lemley, Mark A., Antitrust and the Internet Standardization Problem. Connecticut Law Review, Vol. 28, p. 1041, 1996. Available at SSRN: http://ssrn.com/abstract=44458 or http://dx.doi.org/10.2139/ssrn.44458

[38] See generally Richard Levin et al (1987) and Cohen et al (2000). Using data from the latter survey, Arora et al (2003) find that firms who rate patents as both more important and more effective and tend to do more R&D

[39] James Bessen & Robert M. Hunt, *Software Patent Experiment*, 1 (2004), http://www.researchoninnovation.org/softpat.pdf

According to Prof. Bessen, compared with other patents, software patents tend to be obtained mainly by larger U.S. firms rather then individuals. Moreover, these firms are mostly in manufacturing business and do not, directly, deal with development of software. Most of patents are purchased by "[F]irms in industries that are known to accumulate large patent portfolios and to pursue patents for strategic reasons."[40] It was suggested that firms in these industries may patent heavily in order to obtain strategic advantages, including advantages in negotiations, cross-licensing, blocking competitors, and preventing suits.[41]

II.2.Origin of patentability of software in the U.S. Judicial Interpretation

Patents in U.S. are granted if inventions are useful, new and non-obvious. The third requirement basically means that invention should be more then just conventional or obvious extension of prior art. As a general rule, U.S. patent system does not treat different types of inventions differently.[42] When Congress passed the Patent Act of 1952, it was stated that the new law was to apply to "everything under the sun made by man."[43]

As it was indicated in Chapter I, software programs as such are protected in the U.S. as literary works under Copyright Act of 1972. Subject matter exception for computer programs in became a result of multiple changes in the patent policy. Arrival of Unified

[40] Bessen & Hunt at 3–4

[41] *Id.* at 7

[42] Bessen & Hunt at 3

[43] WILEY, 19 U.S. SENATE REPORT 5 (U.S. Senate, Committee on the Judiciary) (1952), http://ipmall.info/hosted_resources/lipa/patents/Senate_Report_No_1979.pdf

Appeals Court for patent suits in 1982 set up for broader changes in patent law. The court raised the evidentiary standards required to challenge patent validity and tended to broaden the interpretation of patent scope.[44] The court relaxed the standards for evaluating whether or not an invention is obvious to practitioners skilled in the art.[45] The court was also more willing to grant preliminary injunctions to patentees [46]and to sustain large damage awards.[47]. As a result, patents became stronger, more cost-effective and "cheaper" in the meaning that one had to spend less effort to gain it.[48]

As it was elaborated earlier, computer programs were encompassed by US. Copyright Act of 1976 and excluded from patentability *per se*. Before assigning of computer programs under scope of copyright protection, there was extensive research conducted for Congress,[49] as well as decision in *Gottschalk v. Benson* in 1972, where Supreme Court ruled that computer program at hand was a mathematical algorithm and, as such, was excluded from patentability.[50]

However, already in 1981 Supreme Court changed its approach and in *Diamond v. Diehr* ruled that "[A]n invention incorporating computer program could be patented as

[44] Rai 2003, Merges 1997

[45] Cooley 1994, Dunner et al. 1995, Hunt 1999, Lunney 2001

[46] Cunningham 1995, Lanjouw and Lerner 2001

[47] Merges 1997, Kortum and Lerner 1999

[48] Bessen & Hunt

[49] UNITED STATES. CONGRESS. SENATE. JUDICIARY, AN ECONOMIC REVIEW OF THE PATENT SYSTEM: COMMITTEE PRINT...85-2 (1958)

[50] 409 U.S. 63 (1972)

long as the new and non-obvious aspects of the invention did not consist entirely in software."[51] This decision was interpreted to sustain patentability of software.

II.3.Defensive patents and patent thicket.

In comparison with patents in other fields, software patents more often acquired by firms then individuals. Moreover, small and medium business prefers to stick to copyright protection and deems it enough to secure their rights because pursuing patents incur additional costs, consume time and efforts of the "brains" of a company. The trend to increase portfolio of patents develops when business grows. Patent may be used for protection against bigger competitors or deemed as an asset to attract investors. Furthermore, big companies tend to sustain and increase their patent portfolios as a defence means or as exchange object to obtain a cross-license from competitors.

Intensive growth of the industry also attracts "derivative" players, called by Eric Von Hippel, a professor of technological innovation at M.I.T.'s Sloan School of Management, "Patent trolls". The term describes companies or people who gain patents they have no intention of using, the main purpose is to wait for someone else to become successful with similar technology, and then sue them. On the other hand, Intellectual Ventures, an example of patent holding company, argues that what they are trying to create a capital market for inventions similar to venture capital market. Small software firms would be able to get support from venture capitalists and access to private equity market.[52]

[51] 450 U.S. 175 (1981)

[52] *The Big Idea: Funding Eureka!*, HARVARD BUSINESS REVIEW, , http://hbr.org/2010/03/the-big-idea-funding-eureka/ar/1 (last visited Mar 29, 2012)

II.4.Prior art research

To obtain a patent, inventor has to file an application with the patent office. Despite the fact that USPTO has a system for classifying patents "it does not distinguish whether the underlying technology is software or something else."[53] Therefore there is no official definition of software patent in and there are no patent classes for software *per se*.

USPTO assigns software to categories designated for hardware inventions. Since decision in *Diamond v. Diehr,* patent applications are drafted in such a manner that they do not look like patents for software Attorneys avoid draining claims for "pure" software patents, that is, that invention is fully allocated in software.

In absence of classification within system of USPTO and vague language of applications it is difficult to determine which patents were granted for software. Scholars struggled to come up with techniques to identify such patents. For example, adopt their own definition that software patent involves "logic algorithm for processing data that is implemented via stored instructions."[54] The researches basically read through patents granted by USPTO to reveal software patents.

Further the data was applied to find out how software patents influence R&D in the firms that obtain such patents and identify if it was beneficial for the industry. According to the results of the research - firms, which opted to patenting their software, invest less into

[53] Bessen & Hunt at 7

[54] *Id*. at 8

their R&D. The conclusion is that patents consume firm's resources and may eventually lead to industry decline.

Not only firms spend funds for increase their patent thickets, quite probably, with trivial patents, but also threat smaller businesses with litigation should such business develop similar or competitive software. Small businesses prefer to abandon development rather then involve into expensive and lengthy litigation. Strong patent portfolio scares off smaller developers. They simply lack funds to question patent validity in court.

One of the conventional techniques to prevent issuance of trivial patents is to research prior art in the field. With respect to software, a patent office may encounter a number of challenges:

- Novelty of the field
- Lack of expertise
- Absence of classification within patent system
- Vague language of applications
- Global nature of software programs

Given these obstacles, scholars are convinced that USPTO has granted numerous trivial patents for standard, customary, open-sourced or obvious software.

Programmer's freedom of design choice is often restricted by outside considerations such as (1) the mechanical specifications of the computer on which the particular program is intended to run; (2) compatibility requirements of other programs with which a program is designed to operate in conjunction; (3) computer manufacturer's design

standards; (4) demands of the industry being services; (5) widely accepted programming practices within computer industry.[55]As a result of such restrictions sometimes there is an exhaustive number of solutions to perform certain operations. If such solution was patented due to poor prior art research, a developer is precluded from using it, though it may be customary in the field.

Being aware of weak prior art research and growing number of trivial patents, USPTO has launched a Peertopatent joint project with New York law School. The aim of the project is help USPTO improve system of patent issuance through assessment of claims pending for patent. USPTO has opened patent examination process to the public. Outside experts are invited to submit prior art references. Thus, community contributes with relevant information about prior art regarding a pending patent and patent examiner makes a decision relying on the submitted information and legal standards.[56]Indeed, it can contribute greatly to creation of healthier and more competitive environment in the field and effectively eliminate trivial patents. Independent programmers' community is distinguished as very active in defending free sharing of knowledge, especially open source movement and enthusiastically undertakes to notify the Patent Office about if there is claim for prior art.

Despite the fact that the U.S. is example of successful software industry, many experts in the field raise the question of possible decline due to strong patentability practice.

[55] Lemley, at 49

[56] http://peertopatent.org/

Even Patent Office has become concerned with their practice and initiated collaboration with the public to avoid issuance of trivial patents.

II.5.European patent office and battle against patentability of software in Europe

U.S. has experienced accelerated growth in software industry and such success was mainly attributed to availability of patents for computer programs. Conventional approach is that patents provide businesses and individuals with more incentive granting strong protection and its enforcement under patent law. Firms argue that it is patent behind strong motivation to engage in inventive activity as demonstrated by the U.S. However, it should be noted that Fritz Machlup in a Report delivered to Congress, came up with a conclusion that:

> "None of the empirical evidence at our disposal and none of the theoretical arguments presented either confirms or confutes the belief that the patent system has promoted the progress of the technical arts and the productivity of the economy."[57]

As it was mentioned above, the European Patent Office has already engaged into limited patentability of software under to concept of 'computer-implemented invention'. Official recognition of software patentability would be very much welcomed by big firms in Europe Thought the Directive was rejected, the discussion continuous and more patents are being granted by EPO.

[57] JUDICIARY, An economic review of the patent system: Committee print...85-2

Patents do offer desirable degree of protection for software sought by firms and developers. They justify it by incentive theory and motivation to invest into R&D. On the other hand. One of considerations to be taken into account is life cycle of a program. Patents are granted for twenty years. There is little data from authoritative source available to analyze how long does software program "lives". It is true that particular elements of the code are still present from the "dawn" times of the industry, however it might be useful to find out if software need twenty years of protection. One would say that with fast and dynamic development, coupled with relatively cheap means of "production" render twenty years as highly unnecessary.[58] One of the major conclusions of the Chapter is that patents and the patent system will or will not halt development and innovation, mainly depending on how patents are used, whether for defensive purposes or to gain dominant position on the market, to threaten competitors or as an asset for attraction of investors.

[58] Note of author's opinion - access to open source and access to relatively cheap hardware empowers practically anyone to be programmer. Availability of "production means" to common people is considered to be one of the reasons that software developed so quickly.

Chapter III - Protection of software programs under copyright

Legislation usually maintains a "reactive" position and enters to regulate issues when they already came into existence and require provisions to coordinate relations of all involved parties. Software emerged as exclusive and highly limited phenomena. It went hand in hand with cumbersome hardware and considered to be its integral part. Initiative to develop programming flourished mainly in academic and university environments. Students and teachers toyed with basic programming languages and created first algorithms to perform simple functions. First source codes were shared freely or for nominal price among developers. When it became obvious that software poses commercial interest law had to step in. At that point a considerable number of programs had been developed thus contributing to prior art.

III.1.Scope of protection of software under copyright. Background.

The written nature of the source code and the fact that it is human-readable defined that it will be assigned for protection under copyright as literary work.[59] Since a fundamental principle underlying copyright law is that only expression of the ideas, and not the ideas themselves, are entitled to copyright protection, it is prohibited to make an exact copy of software's source code and object code.[60] Copyright protection is granted automatically at the moment a work is fixed on a tangible medium. As it is only expression that is protected under copyright, it is impossible to prohibit someone from developing similar program – copy the idea.

[59] Berne Convention, TRIPS agreement, U.S. Copyright Act

[60] Source code - written instructions for a program readable by humans; object code - written instructions for a program readable by computer , see Lemley at 1115

Apple v Franklin Computer Corp. established that exact copying of computer program code was copyright infringement.[61] Nevertheless, without resorting to direct copying of the source code but rather employing the idea, programmers could afford to develop similar programs.

Due to the fact that initially software was sold as part of hardware, it was the manufacturers of computers who questioned appropriateness of copyright protection. Manufacturers realized that development of software was profitable because they could sell it for additional revenues by addressing needs of individual customers, who required special programs. However with copyright, they could not prevent programmers from developing alternative software. Such software performed similar functions and could run on the platform of computers but was written in a different code. Consequently, customers had an option whether to purchase software of manufacturers or use alternative that could be cheaper.

III.2.Reverse engineering

As source code is human-readable, it enables programmers to interpret and modify it. Most commercial software developers provide object code to their customers, which is only machine-readable, therefore prevent or at least limit independent development through reverse engineering, modifications or interpretations. Discovery by reverse engineering that is, by starting with the known product and working backward to find the method by which it was developed, [62] is one of the techniques used to 'clone' the idea.

[61] 714 F.2d 1240 (3d Cir. 1983), Lemley at 39.

[62] Lemley at 27

Programmer reveals how a program functions and then writes a clone program in a different programming language or uses different combination of algorithm sequence to produce the same result. Reverse engineering is treated differently by IP regimes. Patent prohibits reverse engineering if it involves subsequent making, using or selling of the patented invention – strictly speaking the result of the work,[63] while copyright provides fair use defence if there is no direct copying involved. Both copyright and trademark have fair use defence and in the US stand shoulder to shoulder with freedom of expression and socio-political marketplace of ideas. US patent law has no fair use defence and its experimental use defence is only a mirage.[64] Supporters of copyright claim that the fair use defence provides for opportunity to benefit from ideas of developed programs and contribute to further innovation, while copying prohibition is enough to satisfy revenue expectations of developers and motivate them.

Mainly firms that are engaged in manufacturing of goods containing elements of software rather then solely software developers seek to gain patents for their inventions. They offered customers hardware than can function properly only with their own software. Since today software is not necessarily integral part of hardware, independent developers write application programs for users that are run on platforms of system programs of other developers. Such other software developers may seek to prevent competitors to develop software that would be have to run on their platform, e.g.

[63] *Id.* at 125

[64] *The structure of intellectual property law: can one size fit all*? Ed.Annette Kur, Published Cheltenham, UK , (2011) at 40 HE STRUCTURE OF INTELLECTUAL PROPERTY LAW: CAN ONE SIZEFIT ALL? (Edward Elgar) (2011)

Microsoft which sought to limit customers in using other Internet browsers or media players (application programs) then those offered by Windows platform.

Firms or individuals involved solely into software development are bound by computer on which software should run, other software with which it should interact and requirements for particular software solutions therefore they do not always have a wide range of expression. Predetermined aspects of program's design, structure and code set boundaries to 'artistic' expression of programmer.[65] These limitations are used by copyright opponents to state that there is little artistic expression in software and by its nature it is closer to inventions then to literary works

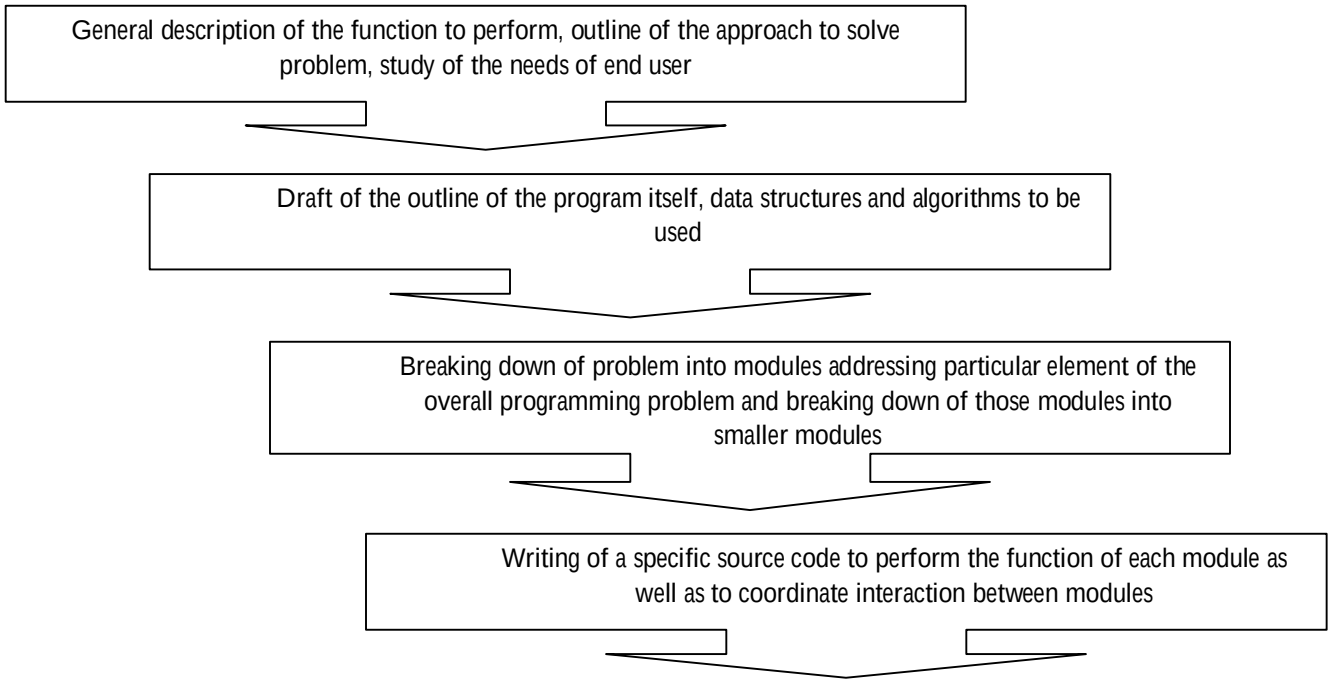
III.3.Judicial interpretation of copyright protection. Idea, process and tangible effect.

As it was pointed out in Chapter II, copyright is also available as protection for software in the U.S. For judicial interpretation it was particularly complicated to distinguish what is software and which elements of it could be copied. In *Whelan Associates v. Jaslow Dental Laboratory* the court has extended copyright protection beyond source and object code and included computer programme's 'structure", "sequence" and "organization" as the two programmes were written in different languages however demonstrated similar outputs, menus, screen displays, etc. Such overbroad inclusion and "amorphous" concept as "general structure" coupled with easily obtained copyright

[65] DAVID NIMMER, COPYRIGHT ILLUMINATED: REFOCUSING THE DIFFUSE US STATUTE 508 (Kluwer Law International□; Sold and distributed in North, Central and South America by Aspen Publishers) (2008)

and no disclosure requirement causes a degree of concern of stiffening in the field.[66] Therefore, it is arguable if copyright does not render a monopolistic position with such extensive interpretation at its disposal.

Professor Nimer analyzed court decisions to identify tests that were used by courts to establish copyright infringement. Developing software is nothing like writing a literary work, it is not spontaneous. Usually, software developer would undertake a sequence of steps [fig.2] to come up with necessary software program:



[Fig.2]

While identifying if there is copyright infringement involved, courts have to take into account several aspects. First, there is a merger doctrine, when idea and its expression

[66] *Id.* at 507-509, *Whelan Associates v. Jaslow Dental Laboratory* 797 F.2d 1222, 230 USPQ 481

merge into one. Under the merger doctrine, "[W]hen an idea can be expressed in only one fashion, that expression is not protected by copyright, as the result would be to provide a monopoly over the idea itself."[67] Therefore, courts should deny copyright infringement claims if there is only one solution to a particular problem in software development.

Another doctrine, that is similar to merger doctrine, is *scenes a faire*, which prevents copyright claims if there is an exhaustive list of ways to express something. It was recognized in the context of software in *Q-Co Indus. V. Hoffman*.[68] This doctrine applies particularly to computer programs because in many cases it is "[V]irtually impossible to write a program to perform particular functions in a specific computing environment without employing standard techniques."[69] It is invoked to deny protection to program elements whose inclusion into structure was dictated purely by efficiency concerns.

Despite the fact that there may be numerous ways to address the same problem it is the "[P]ractical consideration [that] considerably limit the number of choices of expression available to the programmer."[70] As indicated, "[T]he computer software industry progresses by stepping stone improvement process, with each innovation building on

DAVID NIMMER, COPYRIGHT ILLUMINATED: REFOCUSING THE DIFFUSE US STATUTE 508 (Kluwer Law International□; Sold and distributed in North, Central and South America by Aspen Publishers) (2008)

[67] Copyright illuminated p 518

[68] 625 F.Supp. 608 (S.D.N.Y. 1985)

[69] NIMMER at 13–65

[70] *Id.* at 509

past innovations to produce […] an improved product".[71] Therefore, referring to merger and *scenes a faire* doctrines courts may narrow copyright protection and exclude elements that became customary in the field. It would remedy to a certain extend overbroad interpretation from *Whelan Associates v. Jaslow Dental Laboratory.*

It is undisputable that hardware-imposed constrains may extend to numerous aspects of program's design[72]:

- Hardware standards (a number of available function key on keyboard, etc);
- Software standards: compatibility with operating system, compatibility with other programs run on the same hardware;
- Computer manufacturers' design Standards: consistent familiar interface (e.g. IBM's Common User Access SAA Manual) to ensure compatibility between personal computers, minicomputers and mainframes which specifies such details as information layout on the screen, color schemes etc.);
- Target Industry practices: Computer industry programming practices (programming practices and techniques that came into traditional use in software industry and rely on a number of traditional solutions to addressing a problem in programming;

[71] Howard Root, *Copyright Infringement of Computer Programs: A Modification of the Substantial Similarity Test*, 68 MINN L. REV. 1264, 1292 (1984).

[72] NIMMER at 522–525

– Elements taken from public domain (programmers will often build existing public domain software into their works.

Merger and *scenes a faire* doctrines, as well as exclusion of standards and customary practices from copyright protection would allow to filter software elements and boil them down to concentration of what should be copyrighted in a particular program.

III.4.Public domain and Open Source

Tradition of sharing and cooperation exists in software development for a long time now, "[B]ut in recent years, both the scale and formalization of the activity have expanded dramatically with the widespread diffusion of the Internet."[73]Open source movement evolved within academic and research facilities and sharing source code was commonplace.[74]

> "Many of the cooperative efforts in 1970 focused on the development of an then installed across institutions, being transferred freely or for a nominal charge."[75]

In 1980's AT&T, in which Bell Laboratories highly successful Unix was developed started enforcing its intellectual property rights. In response open source society took first steps to formalize the rules behind collaborative software development. Richard

[73] ECONOMICS OF INTELLECTUAL PROPERTY LAW AT 256 (Edward Elgar) (2007)

[74] Open Source -Of or related to software that includes human-readable source code and can be freely revised;

[75] ECONOMICS OF INTELLECTUAL PROPERTY LAW at 257

Stallman from MIT Artificial Intelligence Laboratory established Free Software Foundation.

The purpose of the Foundation was to develop and spread a variety of software free of charge. For this purpose the foundation introduced "[F]ormal licensing procedure that aimed to preclude the assertion of patent rights concerning cooperatively developed software."[76] Pursuant to provisions of the General Public Licence (GNL) or "copyleft" one could modify and/or distribute open source software and in exchange had to make their source code freely available (or for some nominal cost) as well as not to impose licensing restrictions on others. Today, according to the data on Open Source initiative website, there are nine popular and widely used open source licences.[77] Such movement to secure open source community rights evidences strong motivation of the contributors to the field to prevent patenting of freely accessible codes.

To a certain extend, it is still unclear which factors motivate programmers to contribute to open source. Programmers bear a number of costs. The time costs incorporate missed an opportunity to gain financial benefit from their work should they undertook commercial project or be disadvantageous to academic or student output and results. However, as it was reviewed by Lerner and Tirole there is 'signalling incentive' that drives open source developers.[78] It includes full initiative and opportunity to demonstrate

[76] ECONOMICS OF INTELLECTUAL PROPERTY LAW *Id.* at 257

[77] http://www.opensource.org/licenses/alphabetical

[78] Tirole, Jean and Lerner, Josh, The Scope of Open Source Licensing (November 2002). Harvard NOM Working Paper No. 02-42. Available at SSRN: http://ssrn.com/abstract=354220 or http://dx.doi.org/10.2139/ssrn.354220

talent and skill, gain respect and recognition among peers as well as reputation and credit for the performed work.

Facts evidence that open source provides commercial opportunities, though may be delayed in time. Open source licensing does not preclude commercial use, it is not against commercial application, and instead it is anti-'lock-in'. Open-source software can be combined with commercial software. Commercial programs can be developed by open source tools, and run on open source software

The revenue model for open source is based on service provision rather then licensing namely: installation, customization and enhancement services to individual clients to meet their specific business needs. Whilst code is indeed open source, and any business could extend it for its individual business purposes, most firms are not inclined to bother with this activity. If it is not their core business, they turn to developers. Therefore, software firms or individuals may raise money solely from rendering services of software customization.

Among others, it is worth mentioning that open source may assist in attainment of venture capital, help launch independent project or involve into open source project of commercial software developer (e.g. Mozilla project of Netscape).

Development of individual application programs require large team work and considerable capital costs as opposed to individual contribution from open source developers and no capital investment. Successful open source projects such as Apache Linux, Perl, Sendmail prove that open source and free sharing do not pose threat to

incentive for further creation. A user who upgrades a program (which is cheap with the open source) will want synchronized upgrades and efficient level of backwards compatibility.[79] For commercial software upgrading and synchronizations means large expenses. For example, Windows programmer devote a lot of time to remedy these issues. Before commercial release of the operation system it takes about three years to fix various application programming interfaces.[80]

To conclude, copyright is beneficial for start-ups and small firms as effectively helps them to enforce their rights without considerable costs. Copyright is automatic therefore time-efficient. It does not require registration as opposed to patent. Copyright fair use defence and reverse engineering are invoked to justify building on previous programs and though not particularly welcomed by some developers benefit to the society in the long run. Nevertheless, it should be pointed out that copyright is not entirely free from monopolistic aspects. Also, it is granted for fifty-seventy years. If we count back, software as such has hardly existed for more then seventy years. This observation leaves me with the same question – What is the life-span of the program? Some argue that it is from five to seven years. Perhaps a good solution would be to grant more stringent but shorter protection to software.

[79] Tirole, Jean and Lerner, Josh, The Scope of Open Source Licensing (November 2002). Harvard NOM Working Paper No. 02-42. Available at SSRN: http://ssrn.com/abstract=354220 or http://dx.doi.org/10.2139/ssrn.354220

[80] Tirole, Jean and Lerner, Josh, The Scope of Open Source Licensing (November 2002). Harvard NOM Working Paper No. 02-42. Available at SSRN: http://ssrn.com/abstract=354220 or http://dx.doi.org/10.2139/ssrn.354220

Chapter IV - Software protection and its impact from economic perspective

The ultimate objective in view of software programs is to balance interests of developers of software programs and society. Developers should get reward for the efforts and incentive for further development. Society should benefit from increased innovation, wider range of available products, access to information to give opportunity to others to develop new products based on prior art. To expand the economic aspect of the problem, I consider economic value of software not only with respect to developers who design it but also include other developers who build on information that is disclosed to public as well as society that benefits from economic growth, innovation and spurred development in the industry.

IV.1.Monopoly and software development

The historical justification of the patents system has very much centred on the importance of the monopoly-like position offered by the patent.[81]It is not unnatural that some developers are inclined to favour monopolistic approach. Enlarging patent protection and precluding public access underlie private interest. Software development within R&D in big firms involves considerable financial and time costs as well as require qualified experts and team work. Developers seek to retain their position with less disclosure and longer protection.

Nobel Laureate Kenneth Arrow examined correlation between competition and incentive to invent. He argues that there is a trade-off between perfect competition and patent

[81] Bessen & Hunt at 3–4, at 63, a generalization from Dam, Kenneth W. (1994), 'The Economic Underpinnings of Patent Law', 23 (1) The Journal of Legal Studies, at 241-71.

protection. In short run, patent protection raises prices from competitive levels and thus misallocates society's resources. But in the long run this higher prices create the very incentives needed for more invention, and, with it, the technological progress and economic growth that benefit society most. The discussion that unfolded attracted a cluster of scholars to come up with their arguments in favour or against "trade-off" theory and resulted in series of justified but disputable arguments for both patent protection and free access to use.[82] A considerable amount of research of private value of patents into single sectors, individual countries and across-courtiers followed. Interviews of corporate executives, measures of R&D expenses, patent activity on the input side, productivity gains, economic growth on the output side, and surveys of from senior executes in the R&D departments of commercial firms were evaluated to come up with the answer if patents do motivate innovation and further investment. The conclusion that followed states that "[T]he prospect of patent protection was typically factor of third or fourth order of importance to R&D decisions, with the exception of the drug industry and perhaps chemicals."[83] On the other hand, surveys into public sector are inconclusive. One study of 29 countries showed that positive correlation in extended patent protection and R&D investment, while another study into 60 countries showed a negative one.

[82] See generally WILLIAM M LANDES, THE ECONOMIC STRUCTURE OF INTELLECTUAL PROPERTY LAW (Harvard University Press) (2003); Edmund W Kitch, *The Nature and Function of the Patent System*, 20 JOURNAL OF LAW AND ECONOMICS 265–90 (1977), http://ideas.repec.org/a/ucp/jlawec/v20y1977i2p265-90.html (last visited Mar 30, 2012)

[83] THE LAW AND ECONOMICS OF PROGRESS: IP RIGHTS AND COMPETITION POLICY 6

In this context it is also important to mention Open Source software developers. As it was argued before, an open source phenomenon undermines incentive theory of patent supporters. Open Source should not be deemed as the panacea to remedy complex situation today as there is no guarantee that developers will succeed with their product, however there is similarly no such guarantee of success with closed-source software while financial investments are a lot higher.

Conclusion

The uncertainty in the software industry has raged for too long and demands to come up with effective solution. Time has shown that formalistic approach to assign software to copyright or patent has proved inefficient. If I summarize main argument pro and contra each legal vehicle I come up with the following:

Patents:

- ✓ Traditional incentive theory
- ✓ Better protection for "idea" of program
- ✓ Effective enforcement
- ✓ No fair use defence
- ✓ Stringent control
- ✓ Asset to attract investors
- ✓ Shorter in time

but

- Requires registration with patent office
- Expensive to obtain
- Time-consuming
- Potentially leads to monopolisation
- Patent thickets
- Outflow of finance from R&D
- Less input into public domain
- Vague legal provision
- High litigation costs

Copyright:

- ✓ Automatic, no registration
- ✓ Free of charge
- ✓ Fair use defence
- ✓ Monopoly is less likely
- ✓ Open source and public domain benefits

but

- Protection only for expression of idea
- Difficult to enforce
- Unnecessary long protection
- Overbroad interpretation is possible
- Potentially easy to abuse and "copy" by reverse engineering

From the above it is deduced that both copyright and patent and be argued as good for software depending on the side that argues.

If it is a commercial developer with strong economic position, patent would be more appealing because developer can afford it. Patent gives stronger control over product, allows precluding others from developing similar technology or ensured bargaining power with competitors to obtain cross-licence. Patent would be also considered as better investment. One never knows if R&D will come up with a new and successful product so it is more logical to enforce position with those products that are already successful.

If we put ourselves into shoes of start-up or small business, it might encounter costs it will be unable to bear. First of all financial costs to obtain patent, secondly time costs. It may take up to three years to get a patent, many start-ups do not live that long. Therefore, copyright is more appealing to them. Copyright grants protection automatically and free of charge. If a start-up develops a successful product, it becomes successful immediately and gives advantage over competitors to gain many customers. Even if alternative software is developed, a firm may raise money from rendering a variety of services or customizing their product. Developers may also benefit from open source and build on the work of others. If software was strictly patentable, the firm or individual developer would have to “reinvent the wheel” every time it would want to develop an independent program or purchase a licence from patent holder, therefore undertake either time or financial costs or both.

However, the worst scenario may occur if developer argues both copyright and patent protection. As legal framework both in the U.S. and Europe allow for statutory automatic copyright protection and patent offices, supported by judicial interpretation, issue

patents. Such practice does signal monopolization in the industry and further chilling of innovation.

Therefore, upon analyzing scope of patent and copyright protection, their positive and negative effects, I come up with a conclusion that the best way to address the need of software industry and safeguard benefit of society from innovations is to develop for EU a separate legal provision outside of patent or copyright umbrella.

This legal provision should define what is software, provide classification, define which parts or elements should be protected, how long and how exactly, it should address question of open source and provide a form of fair use defence to motivate future developments by other programmers, ensure effective enforcement mechanism.

It is under no condition an easy solution. To develop such provision, legislator will have to engage lawyers, economists, developers, scholars and experts from different fields. However, easy solutions of complementary use of patent and copyright proved to be ineffective. Ill-fitting law puts extra burden of costs on the shoulders of software developers and deprives society of prospects for dynamic and effective innovation.

Bibliography

Books

1. RB Bakels, *The patentability of computer programs*, (2002)
2. James Bessen & Robert M. Hunt, *Software Patent Experiment*, (2004), http://www.researchoninnovation.org/softpat.pdf
3. James E. Bessen & Robert M. Hunt, *An Empirical Look at Software Patents*, SSRN ELIBRARY (2004), http://papers.ssrn.com/sol3/papers.cfm?abstract_id=461701&rec=1&srcabs=886905 (last visited Mar 29, 2012)
4. Aaron D. Charfoos, *How Far Have We Come, And Where Do We Go From Here: The Status Of Global Computer Software Protection Under The TRIPS Agreement*, NORTHWESTERN JOURNAL OF INTERNATIONAL LAW AND BUSINESS 1–39 (2002), http://www.kirkland.com/siteFiles/kirkexp/publications/2499/Document1/Charfoos_Northwestern%20Univ%20School%20of%20Law.pdf
5. MARIO CISNEROS, PATENTABILITY REQUIREMENTS FOR NANOTECHNOLOGICAL INVENTIONS: AN APPROACH FROM THE EUROPEAN PATENT CONVENTION PERSPECTIVE (Nomos Verlagsgesellschaft) (2010)
6. BLACK'S LAW DICTIONARY (West 9th ed) (2009)
7. Andrés Guadamuz, *The Software Patent Debate*, SSRN ELIBRARY , http://papers.ssrn.com/sol3/papers.cfm?abstract_id=886905& (last visited Mar 29, 2012)
8. ROBERT HART ET AL., THE ECONOMIC IMPACT OF PATENTABILITY OF COMPUTER PROGRAMS. REPORT TO THE EUROPEAN COMMISSION. (Intellectual Property Institute), http://ec.europa.eu/internal_market/indprop/docs/comp/study_en.pdf

9. UNITED STATES. CONGRESS. SENATE. JUDICIARY, AN ECONOMIC REVIEW OF THE PATENT SYSTEM: COMMITTEE PRINT...85-2 (1958)

10. Edmund W Kitch, *The Nature and Function of the Patent System*, 20 JOURNAL OF LAW AND ECONOMICS 265–90 (1977), http://ideas.repec.org/a/ucp/jlawec/v20y1977i2p265-90.html (last visited Mar 30, 2012)

11. WILLIAM M LANDES, THE ECONOMIC STRUCTURE OF INTELLECTUAL PROPERTY LAW (Harvard University Press) (2003)

12. SOFTWARE AND INTERNET LAW (Aspen Publishers 3rd ed) (2006)

13. Mark A. Lemley, *Antitrust and the Internet Standardization Problem*, SSRN ELIBRARY , http://papers.ssrn.com/sol3/papers.cfm?abstract_id=44458 (last visited Mar 29, 2012)

14. Jacqueline D. Lipton, *IP's Problem Child: Shifting the Paradigms for Software Protection*, SSRN ELIBRARY , http://papers.ssrn.com/sol3/papers.cfm?abstract_id=901604 (last visited Mar 29, 2012)

15. Fritz Machlup & Edith Penrose, *The Patent Controversy in the Nineteenth Century*, 10 THE JOURNAL OF ECONOMIC HISTORY 1–29 (1950)

16. ECONOMICS OF INTELLECTUAL PROPERTY LAW (Edward Elgar) (2007)

17. THE STRUCTURE OF INTELLECTUAL PROPERTY LAW: CAN ONE SIZEFIT ALL? (Edward Elgar) (2011)

18. Andrew Nieh, *Software Wars: The Patent Menace*, 55 NEW YORK LAW SCHOOL LAW REVIEW 295

19. DAVID NIMMER, COPYRIGHT ILLUMINATED: REFOCUSING THE DIFFUSE US STATUTE (Kluwer Law International ; Sold and distributed in North, Central and South America by Aspen Publishers) (2008)

20. PATENT INFORMATICS TEAM, PATENT THICKETS (Intellectual Property Office) (2011), http://www.ipo.gov.uk/informatic-thickets.pdf

21. THE LAW AND ECONOMICS OF PROGRESS: IP RIGHTS AND COMPETITION POLICY

22. Jean Tirole & Josh Lerner, *The Scope of Open Source Licensing*, SSRN ELIBRARY (2002), http://papers.ssrn.com/sol3/papers.cfm?abstract_id=354220 (last visited Mar 29, 2012)

23. WILEY, 19 U.S. SENATE REPORT (U.S. Senate, Committee on the Judiciary) (1952), http://ipmall.info/hosted_resources/lipa/patents/Senate_Report_No_1979.pdf

24. EUROPEAN SOFTWAR PATENT STATISTICS FOUNDATION FOR FREE INFORMATION INFRASTRUCTURE (FFII), http://eupat.ffii.org/patents/stats/index.en.html (last visited Mar 29, 2012)

25. HTTP://MISES.ORG/MEDIA/POSTER/1182 , http://mises.org/document/1182/An-Economic-Review-of-the-Patent-System (last visited Mar 29, 2012)

26. *The Big Idea: Funding Eureka!*, HARVARD BUSINESS REVIEW, , http://hbr.org/2010/03/the-big-idea-funding-eureka/ar/1 (last visited Mar 29, 2012)

Cases

"VICOM" (1987) 2 EPOR 74;

Merrill Lynch's Application (1989) RPCC 561;

Gale (1991) RPC 305

Feist Publ'ns, Inc. v. Rural Tel.Ser. Co., 499 U.S. 340 (1991)

Gottschalk v. Benson, 409 U.S.63.

Diamond v. Diehr, 450 U.S. at 175