THE LEGAL PROTECTION AND USE OF OPEN SOURCE COMPUTER PROGRAMS

by Krzysztof Siewicz

LL.M. LONG THESIS
PROFESSOR: György Boytha, LL.D.
Central European University
1051 Budapest, Nador utca 9.
Hungary

Table of Contents

List of Abbreviations....iii

Executive Summary....v

Introduction....1

Chapter 1 - Legal Protection and Use of Computer Programs in General....5

1.1 Computer Programs and Software Market....5

1.2 Copyright Protection of Computer Programs....8

1.3 Patent Protection of Computer Programs....14

1.4 Trade Secrets Protection of Computer Programs....19

1.5 Contract Law and Licensing of Computer Programs....19

1.6 Theory and Practice of Current System of Legal Protection....21

Chapter 2 - History and Present State of Open Source Software....27

2.1 UNIX....27

2.2 Free Software and Project GNU....30

2.3 Linux....31

2.4 Open Source Initiative....33

2.5 Various Open Source Projects....34

2.5.1 Apache HTTP Server....35

2.5.2 Mozilla Web Browser....36

2.5.3 OpenOffice.org Office Suite....37

Chapter 3 - Role of Law in Development of Open Source Software....38

3.1 Free Software v. Open Source....38

3.2 Hackers' Culture....45

3.3 Open Source as Marketing Model....49

3.4 Model Open Source Licenses....52

3.4.1 GNU General Public License....53

3.4.2 BSD License....55

3.4.3 Apache Software License....56

3.4.4 Mozilla Public License....57

3.4.5 Sun Industry Standards Source License....59

3.5 Intellectual Property Rights Management in Open Source Movement....60

3.6 Evaluation of Role of Law in Development of Open Source Software....63

Chapter 4 - Legal Controversies of Open Source Licensing....66

4.1 Contract Law and Open Source Licensing....66

4.1.1 U.S. Case Law on Software Licensing....68

4.1.2 UCITA and Relevant European Union Directives....73

4.1.3 Applicability of Sales of Goods Laws....75

4.1.4 Implications for Open Source Licensing....79
4.2 Copyright Law and Open Source Licensing....81
4.2.1 Preemption....82
4.2.2 Copyright Misuse....90
4.2.3 Moral Rights....93
4.3 Possible Consequences of License Invalidity....94
4.3.1 First Sale....95
4.3.2 Fair Use....96
4.3.3 Implied Licenses....97
4.3.4 Liability for Open Source Software....100
4.4 Implications of "Copyleft" Infringement....104
4.5 Impact of Software Patents on Open Source Movement....105
4.6 Evaluation of Legal Controversies of Open Source Licensing....108
Conclusion....112
Bibliography....116
Table of Cases....123

List of Abbreviations

Apache License	Apache Software License, Version 2.0
Art.	Article
Berne Convention	Berne Convention for the Protection of Literary and Artistic Works (1886, Paris Act of 1971)
BSD	Berkeley Software Distribution
BSDI	Berkeley Software Design, Inc.
CISG	UN Convention on Contracts for the International Sale of Goods (Vienna Convention, 1980)
cl.	clause
CLA	Individual Contributor License Agreement (Apache)
CPU	Central Processing Unit
E-Commerce Directive	Directive on certain legal aspects of information society services, in particular electronic commerce, in the Internal Market (E-Commerce Directive), 2000/31/EC, OJ L 178, 17/07/2000, P. 1
EPC	European Patent Convention (1973)
EPO	European Patent Office
et al.	*et alia* (and others)
et seq.	*et sequens* (and following)
etc.	*et cetera* (and so on)
EU	European Union
EULA	End-User License Agreement
European Software Directive	Directive on the legal protection of computer programs, 91/250/EEC, OJ L122, 17/05/1991 P. 42
FAQ	Frequently Asked Questions
FFII	The Foundation for Free Information Infrastructure
FSD	Free Software Definition
FSF	Free Software Foundation, Inc.
FTP	File Transfer Protocol
GNU	GNU is Not Unix
GNU GPL	GNU General Public License, Version 2.0
GNU LGPL	GNU Lesser General Public License, Version 2.1
HTTP	Hyper-Text Transfer Protocol
Id.	*Idem* (thereto)

JCA	OpenOffice.org Open Source Project Joint Copyright Assignment by Contributor
MIT Labs	Massachusetts Institute of Technology, Artificial Intelligence Laboratories
MPL	Mozilla Public License, Version 1.1
NCSA	National Center for Supercomputing Applications
No.	Number
OJ	Official Journal
OSD	Open Source Definition
OSI	Open Source Initiative
OSRM	Open Source Risk Management
R&D	Research and Development
RAM	Random Access Memory
Rome Convention	Convention on the Law Applicable to Contractual Obligations (1980)
Sec.	Section
SISSL	Sun Industry Standards Source License, Version 1.1
Software Patents Directive	Proposal for the Directive on the patentability of computer-implemented inventions, 6580/02 PI 10 CODEC 242
TRIPS	WTO Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS) (1994)
U.S.	The United States of America
UCC	Uniform Commercial Code
UCITA	Uniform Computer Information Transactions Act
UK	The United Kingdom
UNCITRAL	United Nations Commission on International Trade Law
Unfair Terms Directive	Directive on unfair terms in consumer contracts, 93/13/EEC, OJ L 095, 21/04/1993 P. 29
USL	Unix System Laboratories
Vol.	Volume
WIPO	World Intellectual Property Organization
WTO	World Trade Organization

Executive Summary

The paper deals with the question of the relation between Open Source computer programs and law on both the theoretical and practical level. The methodology used was to examine legal aspects of computer programs in general in order to estimate the specific issues triggered by Open Source Software in this area. Technical, economic and social aspects of software production were also analyzed. The research encompassed primary and secondary sources of law of the U.S. and EU, non-legal authoritative writings and consultations with experts.

The main areas discussed, in order of appearance, are the general means of protecting computer programs, the theoretical grounds for this system and the ways it is used in practice. Then, the history and some details of technical side of Open Source computer programs is presented together with the social and economic aspects of Open Source Movement. All this research is put together and the essence of the legal system designed by the movement is discussed against the background of general protection. In the end, this system is analyzed by confronting its institutes with the laws of particular jurisdictions.

The key finding of the paper is that the legal protection and use of Open Source computer programs form an innovative and remarkable system, which is in-line with the theory of intellectual property law even though its use of the law is the opposite of the trend known as "proprietary licensing". Although most of the Open Source legal innovations are generally valid and enforceable, much depends on the interpretation given in particular jurisdictions. The finding of the detailed analysis is that there is a need of more diligence in arranging legal relationships between parties involved in the creation and use of Open Source Software.

INTRODUCTION

The development of Open Source computer programs has gained much attention of all the players in the software market. Not only individual users and programmers but also major software development and hardware manufacturing companies together with public administration worldwide are seeking ways to take advantage of this phenomenon and to use it for their diverse ends. Indeed, it proves to be an extraordinary way of organizing the production in the Information Age, having its own rich history, underlying ideology and social institutions. But what is of prior importance for the purposes of this paper, Open Source development has been based on a specific legal system, the features of which deserve special analysis against the background of the legal framework for the protection of software the lawyers have been used to.

Apart from a plethora of pragmatic arguments,[1] which make the legal protection and the use of Open Source computer programs worth researching and producing academic dissertations, thus contributing to its general understanding and legal certainty, there are even more appealing theoretical questions posed by this social phenomenon. First and foremost, all the innovations in the field of law must face the question of their validity and enforceability. This issue has come under the attention of some legal scholars[2] and has even been put before the courts in a few jurisdictions recently.[3] Thus, there are more and more sources identifying the possible legal consequences of developing software under Open Source model, which by now include issues of copyright, contract or constitutional law, but their list and final

[1] Various Open Source Software products hold a significant market share. The scale and organization of Open Source Movement is an outstanding social phenomenon. Various leading companies worldwide contribute or participate in Open Source production. National governments and international institutions promote the development of Open Source Software.
[2] See *e.g.* Moglen, FN 120 and 121, Rosen, FN 213 or Metzger & Jaeger, FN 229.
[3] See FN 155.

interpretation is far from being complete. Apart from the need to add to this detailed analysis, the legal research of Open Source development requires to take a look at it from a distance and ask some important general questions about the system for the protection of intellectual property. These issues have been considered to the greatest extent by the originators of Open Source Movement,[4] mostly non-lawyers, but some prominent legal scholars[5] have also been addressing the matter. Here, the debate may be considered even more heated and far from being resolved.

The purpose of this paper is to contribute to both of these discussion threads. Namely, some of the most interesting and crucial legal issues concerning particular Open Source projects and their licenses are looked upon in order to evaluate the impact of the current legal system on their development. At the same time the paper analyzes how Open Source phenomenon influences the theory of intellectual property protection. Thus, the purpose is to observe, analyze and comment on the reciprocal relation between Open Source Movement and law.

The position to the topic is that the legal system designed to protect and allow for the use of Open Source computer programs complies with the theory of intellectual property laws. Another major finding of this paper is that given the success of Open Source Movement, the law as used by them is sufficient and calls for stronger protection should not be followed blindly. As to the specific findings, they may be summarized that black letter laws protect the legal model designed by Open Source licensors by holding it valid and enforceable to a great extent. There are some reservations flowing from the fact that the participants in Open Source Movement use the law in a specific way, which does not always

[4] See *e.g.* DiBona *et al*, FN 65, Stallman, FN 92 or Perens, FN 254.
[5] See *e.g.* Lessig, FN 65.

comply with particular regulations and should therefore exercise more diligence in this area. These specific findings encompass issues related to contract formation, the scope of rights and obligations of the users and liability of licensors.

The methodology of the research for this paper was as follows. Some studies of the technical, economic and social background for Open Source Movement were conducted in order to understand the subject matter better, realize how exactly it differs from other software production models and what are the special features of its end-products. Moreover, the legal system for the protection of software in general was analyzed for the same purpose of identifying particular Open Source-specific issues. The research was conducted on the primary and secondary sources of law of the two big economies of the World – the U.S. and EU as well as the writings of various non-legal practitioners. Diverse methods of collecting research material were used, such as searching the libraries, Internet or consulting with legal and technical experts.

In the course of the research, the following structure of the paper was elaborated. The first Chapter provides the necessary background by presenting the general rules for the protection and use of computer programs. Since Open Source has been much in the opposition to the system that in this paper is referred to as "proprietary", this model is also described in the first Chapter. Thus, it shows what the starting point is before deciding whether to opt for the proprietary or Open Source approach, the rationale of those who choose the former and the effect of such decision. The second Chapter presents the history and the actual state of Open Source Software, in order to familiarize the reader with basic terms and names used extensively later on. The purpose of the third Chapter is to estimate the role of law in the development of Open Source Software. It discusses the rationale and ideology of all the parties to Open Source Movement and explains what interests are at stake, what theories are used or are likely to be used to support them. Thus, the second and third Chapters together show what exactly makes the difference between proprietary and Open Source software and also confront the latter with the theory of intellectual property laws. The fourth Chapter contains detailed analysis of various legal controversies resulting from Open Source Software development, as its purpose is to find out whether the law can indeed enforce the system for the protection designed by Open Source licensors. This issue is also analyzed against the background of general legal system for the protection of computer programs.

Chapter 1 - Legal Protection and Use of Computer Programs in General

1.1 Computer Programs and Software Market

A computer program is an algorithm, that is a set of instructions serving the purpose of solving a given task, expressed in one of the programming languages understandable to a computer and capable of being executed by it. This is not a legal definition and the understanding of computer program in legal systems varies. Some national laws do not include any definition, thus forcing courts and other institutions to rely on the current development of the computer sciences. On the other hand, some definitions are designed in a quite elaborate way, such as the one from the WIPO Model Provisions stating that computer program is

> a set of instructions, which can, once transcribed on a medium decodable by a machine, make indicate, accomplish or obtain a particular function, task or result by a machine capable of treating information.[6]

Sec. 101 of the U.S. Copyright Act defines 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".[7] There is no definition in the Directive on the legal protection of computer programs (European Software Directive)[8] but the Directive Proposal intended to encompass by the word "program" "the expression in any form, language, notation or code, of a set of instructions the purpose of which is to cause a computer to execute a particular task or function".[9]

Computer programs may be expressed, generally speaking, in two forms: source

6 WIPO Model Provisions on the Protection of Computer Software (1978).

7 Copyright Act Sec. 101, 17 U.S.C. Sec. 101 (1976, as amended).

8 91/250/EEC, OJ L122, 17/05/1991 P. 42.

9 COM (88) 816 final – SYN 183 [1989] OJ C91/9, as cited by: Estelle Derclaye, *Software Copyright Protection: Can Europe Learn from American Case Law?* Part 1, 1 European Intellectual Property Review 10 (2000).

code and object code. A source code is a transcript in one of the high-level programming languages, such as C or Java. These languages are understandable to human beings and are used for writing and modifying (debugging, updating) programs. An object code, also referred to as a binary, is a translation of source code into a machine-readable string of 0s and 1s, prepared with the help of special compiler programs. The object code is practically impossible to be understood by a human; but, as computers directly understand only the object code, programs must always be translated (compiled) into it in the end. When the program is finished it is possible to distribute it in the object code only, as it is sufficient for the purpose of running it on a computer.

A computer program, expressed in either of the two forms of code, together with some additional elements constitutes software, as opposed to hardware, usually referring to tangible electronic equipment. These elements generally fall into one of the following categories: (1) software development tools; (2) preparatory design material; (3) input data; (4) program output; (5) additional materials (*e.g.* manuals, help modules); (6) interfaces (including, but not limited to graphical on-screen presentation, the "look and feel" of program).[10] Similarly to the source code, software development tools and preparatory design material are important only during the creation of the program, thus they are usually not marketed together with it. Conversely, last four elements, together with the program in object code usually form software in the sense of product that is being offered to end users.

Software differs from traditional tangible goods in a number of ways, apart from the most obvious difference that it is pure information and the form of its fixation is not so much important. For example, it is usually not expected from software to work properly as a

[10] Compare with: DAVID BAINBRIDGE, SOFTWARE COPYRIGHT LAW, 1-3 (Butterworths, London, Edinburgh, Dublin, 4th ed., 1999).

finished end-product. It is common for it to contain mistakes (bugs) resulting from the fact that it is virtually impossible even for most diligent developers to predict all situations that may occur in the computer during the use of program. Usually, after the official release of software and appearance of bugs, additional files (patches) are introduced that fix these mistakes. It is also common for developers to work constantly on and market new versions (upgrades) of software, which extend its functionability. This is the result of a rapid development in all the fields of Information Technology together with the ideological approach in the industry, which does not require working to achieve any final result but rather to constantly improve. Consequently, "the life" of a particular computer program is extremely short (especially in comparison to the length of the copyright or patent protection term) and new versions are introduced in short cycles.

Additionally, the software market is susceptible of "network effects". Because of technical reasons such as compatibility and interoperability requirements, together with simple habituation of users to certain solutions, the value of a computer program as perceived by its users increases every time when additional users choose this particular program.[11] With the increase of the number of users, switching costs raise. In the end, they may prevent consumers from abandoning one product and its vendor even if others are objectively better.

Another characteristic of computer programs and software is that although they are quite expensive to create, they can be multiplied and distributed at practically no costs. This is a feature common to virtually all types of intellectual property, but in the case of software it becomes extremely apparent. The process of creation is lengthy, involves a lot of testing, rewriting and, especially in the case of special-purpose software, consultation with experts in

[11] F. Warren-Boulton *et. al.*, *Economics of intellectual property protection for software: The proper role for copyright*, 3 No. 2 STANDARD VIEW 68-78 (June 1995).

many fields. However, thanks to digital technology and because programs are by their very nature digital, not only can they be copied instantly with the use of minimal expertise and equipment but such copies are practically indistinguishable from originals. Thus, uncontrolled copying would allow many people to benefit from the use of practically the same program, without detracting from this use owners of original. Such free-riding is indeed extremely difficult to control. Obviously, the fact that programs written with a lot of effort of their authors could be copied and used free of charge by unauthorized free-riders, who would receive the same quality product as individuals paying the full fee, has had a strong influence on the system of software legal protection.[12]

This system consists mainly of copyright law. To some extent trade secrets protection is used and, especially recently, patent protection is gaining growing interest. Software licensing agreements form the most significant element of the system, giving rise not only to copyright but also to many contract law concerns. All these will be separately described in the following sections to give the reader a general overview of software legal issues before discussing matters specific to Open Source.

1.2 Copyright Protection of Computer Programs

One of the major benefits of copyright law is the protection against unauthorized copying. However, it was not clear from the very beginning whether computer programs are copyrightable. Out of many features distinguishing programs from works traditionally covered by copyright law two can be brought under consideration, as pointed out by Drexl: utilitarian character and the lack of communicative purpose.[13] Computer programs do not

[12] It should be kept in mind that apart from the legal protection, software distributors widely use technical means that physically prevent unauthorized copying or other unwelcome access of the users.

[13] JOSEPH DREXL, WHAT IS PROTECTED IN A COMPUTER PROGRAM?: COPYRIGHT PROTECTION IN THE UNITED STATES AND EUROPE, 9-12 (VCH Verlagsgesellschaft mbH, Weinheim, New York, 1994).

serve an aesthetic or even educational purpose. They are intended to be just a set of instructions serving an useful (utilitarian) end of accomplishing a task.[14] Moreover, although computer programs may be understood by skilled individuals, they are not written in order to communicate with humans. Their real purpose is to make computers produce certain results. The result of a program, being referred to by some prominent authors as its "behavior",[15] is the source of its value. The same result may be accomplished by many different expressions of program (source and object codes); thus, a particular expression of the program is not so important as it is the case with traditionally copyrighted works.

Although these and some other special features may be considered in the discussion whether copyright law is fit for the protection of computer programs, neither the purpose of a work nor the reason why it is considered valuable have to be analyzed in order to decide whether it is copyrightable. In fact, there are numerous utilitarian works that have benefited from copyright protection long before computer programs. Copyright laws, generally speaking, only require the work to be original and this standard is usually set at a very low level, such as the one given in American case *Feist Publications, Inc. v. Rural Telephone Service Co., Inc.*: "[o]riginal means ... that the work was independently created by the author ... and that it possesses at least some minimal degree of creativity".[16] In Europe, according to the Software Directive the program is original if it is the author's own intellectual creation. This definition, although vague and subject to different interpretations by Member States, in any case is believed to be in-line with the American "minimal degree of

[14] It is true even with regard to game or educational software, as computer programs are not a game or education themselves, they just constitute an algorithm for a computer to provide fun or information to the end users.

[15] Pamela Samuelson *et. al.*, *A Manifesto Concerning The Legal Protection Of Computer Programs*, Columbia Law Review, December 1994, at 2308, 2316 *et seq.*

[16] *Feist Publications, Inc. v. Rural Telephone Service Co., Inc.* 111 S. Ct 1282, 18 U.S.P.Q 2d 1275 (1991).

creativity" test of *Feist*.[17] Therefore, one may say that virtually all computer programs can easily fulfill copyrightability requirements.[18]

Computer programs are subject to copyright protection guarded by such international treaties as the WIPO Copyright Treaty (1996) or Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS) (1994). The former protects computer programs as literary works within the meaning of Art. 2 of the Berne Convention for the Protection of Literary and Artistic Works (Berne Convention), regardless of their mode or form of their expression.[19] Similarly, TRIPS Art. 10.1 obliges its signatories to protect computer programs as literary works under the Berne Convention, whether they are in the source or the object code. This express wording may be interpreted as the effect of discussion, whether the object code, usually prepared by compiler programs without any human interference is an expression eligible for protection.

At the national level, although a *sui generis* protection was considered at first, in the end the existing framework of copyright law was chosen. The U.S. amended its Copyright Act in 1980 to extend the protection to computer programs followed shortly by Japan and the European Software Directive later on in 1991. Therefore, the international agreements just mentioned confirmed only the practice of most advanced economies and reassured common standards for the rest of the world.

In the U.S. the protection of computer programs extends both to the source code (by "statements" and "indirect use" wording of Copyright Act Sec. 101 definition) and the object

[17] Estelle Derclaye, *Software Copyright Protection: Can Europe Learn from American Case Law?* Part 1, 1 EUROPEAN INTELLECTUAL PROPERTY REVIEW 10, 15-16 (2000).
[18] BAINBRIDGE, FN 10 at 12.
[19] Art. 4 of the WIPO Copyright Treaty. The Treaty is a special agreement within the meaning of Art. 20 of the Berne Convention.

code (by “instructions” and “direct use” wording).[20] The scope of the copyright protection is limited by Sec. 102(b), which excludes ideas, procedures, processes, systems, methods of operation, concepts, principles or discoveries. There is; however, no positive enumeration of protected elements and the general copyright law rule applies extending protection to expressions of ideas (forms in which authors create their work), not to the ideas themselves.[21]

There are two main categories of elements of computer programs: literal and non-literal. The copyright protection of literal elements, the textual expression of the program in the form of source or object code has been admitted by the U.S. courts without much ado.[22] As to non-literal elements, such as programs’ structure, sequence, organization, screen displays, general flow charts, and menu structures, the courts have designed four distinct tests all based on the “idea-expression dichotomy”.[23] Applied to computer programs, they result in granting protection to non-literal elements as long as they can be regarded expressions of ideas, not the ideas themselves.[24]

It is believed that the “abstraction-filtration-comparison” test developed in *Computer Associates, Inc. v. Altai, Inc.*[25] takes precedence, although none of the previous

[20] Derclaye, FN 17 at 11.

[21] Diane Rowland, Elizabeth Macdonald, Information Technology Law, 29 (Cavendish Publishing Ltd, London, Sydney, 2nd ed., 1997).

[22] *Williams Electronics, Inc. v. Artic International, Inc.* 685 F.2d 870 (3d Cir. 1982); *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.* 797 F 2d 1222 (3d Cir. 1987); *CMS Software Design Sys., Inc. v. Info Designs, Inc.*, 785 F.2d 1246 (5th Cir. 1986); *Apple Computer, Inc. v. Franklin Computer Corp.*, 714 F.2d 1240 (3d Cir. 1983), cert. dismissed, 464 U.S. 1033 (1984)

[23] Julian Velasco, *The copyrightability of non-literal elements of computer programs*, Columbia Law Review, January 1994, at 242 (citing: *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.* 797 F 2d 1222 (3d Cir. 1987); *Lotus Development Corp. v. Paperback Software Int.* 740 F. Supp. 37 (D. Mass. 1990); *Brown Bag Software, Inc. v. Symanthec Corp.* 960 F 2d 1465 (9th Cir 1992); *Computer Associates, Inc. v. Altai, Inc.* 982 F 2d 693 (2nd Cir. 1992)). There are commentators; however, who see in some of these cases an evolutionary development of the same test. See Arthur R. Miller, *Copyright protection for computer programs, databases, and computer-generated works: is anything new since CONTU?*, 106 Harvard Law Review 977 (1993).

[24] Velasco, FN 23 at 253.

[25] 982 F 2d 693, 706 (2nd Cir 1992).

tests has been expressly overruled so far.[26] The abstraction phase consists of breaking down the scrutinized program into its levels of abstraction.[27] Then, each part is “filtered” separately in the search for copyrightable material by taking away incorporated ideas, incidental expression (merger, scenes a faire), and the public domain.[28] What is left is compared, in the comparison phase, with the structure of the program in order to determine substantial similarities.[29] The court examines not only whether the defendant copied a protected expression, but also takes under account the relative importance of the copied element in relation to the program.[30]

The *Altai* test is by far most detailed and sophisticated. It is also a very strict one, which effects in not granting copyright protection for non-literal elements in every case. Applying the *Altai* criteria, much of them may be copied to another program and still not amount to copyright infringement. However, the *Altai* court noted that remedies may be sought in trade secret law – “an appropriate means by which to secure compensation for software espionage”[31] or patents “with [their] exacting up-front novelty and non-obviousness requirements, [which] might be the more appropriate rubric of protection.”[32]

Copyright protection for computer programs in Europe is granted in national laws. In 1991, the European Communities introduced European Software Directive, which requires in Art. 1.1 that the member states protect computer programs as literary works within the meaning of Berne Convention. Similarly to the U.S. Copyright Act, the directive also relies

[26] Velasco, FN 23 at 243; D. S. Karjala, *Recent United States and International Development in Software Protection*, Part 1, 1 European Intellectual Property Review 13 (1994), Part 2, 2 European Intellectual Property Review 58 (1994).
[27] 982 F 2d 693, 707 (2nd Cir 1992).
[28] *Id.* at 706.
[29] *Id.*
[30] *Id.* at 710.
[31] *Id.* at 721.
[32] *Id.*

on idea-expression dichotomy and its Art. 1.2 excludes from protection “[i]deas and principles which underlie any element of a computer program, including those which underlie its interfaces”. Recital 14 adds “to the extent that logic, algorithms and programming languages comprise ideas and principles, those ideas and principles are not protected”. There is no elaborate exclusion, such as Sec. 102(b) of the Copyright Act but the directive explicitly mentions preparatory design material (Art. 1.1) and codes (Recital 7) as protected. According to Derclaye, the directive also grants protection to user interfaces, though impliedly, in Recital 10 and the Proposal referred to sub-programs routines and modules as protected independently.[33] The directive clarifies further, that the protection is granted regardless of form in which programs are expressed.[34] Therefore, all literal elements (both source and object codes) are protected. Non-literal elements can receive protection only to the extent that they finally permit the building of a computer program.[35] Thus, their protection may be, like in the U.S., regarded as relatively weak.

The questions of exact scope of copyright protection, whether it covers non-literal elements of program or other items constituting software, though still not finally resolved, are not specific with regard to the very subject of this paper. It is only important to have in mind that copyrightability of Open Source programs involves all these abovementioned considerations. On the other hand, reverse engineering and decompilation are usually discussed among most important issues concerning the scope of copyright protection of computer programs. These are techniques used, for example, to assure compatibility and interoperability of a program being developed with other programs that are going to be used

[33] Derclaye, FN 17 at 14.

[34] Recital 7: “... any form, including those which are incorporated into hardware; ... also ... preparatory design work ...”; Art 1.1 repeats the inclusion of preparatory design material.

[35] Derclaye, FN 17 at 11 (relying on the wording of Recital 7).

in the same computer. The extent to which non-copyright holders are allowed to reverse engineer or decompile programs does not have to be addressed in this paper. It is true that one may attempt to decompile an Open Source computer program, for example to make sure it indeed comes from the source code supplied together with it, but this is not such a dividing issue as with regard to proprietary programs.

1.3 Patent Protection of Computer Programs

In comparison with copyright, patent protection seems to be more beneficial towards the intellectual property holder. Copyright, especially in the light of difficulties in extending it over non-literal elements, does not fully protect the value of software, most of which lies not in a particular expression used in the code of computer program but in what it does and in its embodied idea. Thus, attempts to secure patent protection for software are understandable, as patents are tailored to protect functionality. Moreover, as pointed out by Nichols, two main advantages of patent over copyright are that the former protects the holder against similar or even equivalent (interchangeable) inventions and additionally against independent creations because it allows preventing others from using patented products even if made completely on their own.[36] Copyright protects expression (form of the work), whereas patent may be granted for the practical application of innovative ideas effecting in a product or process, so they were traditionally perceived as excluding each other.[37] Odd as it is for other types of works covered by copyright, computer programs may quite often additionally indirectly benefit from patent protection, to the extent they form a part of a product or process claimed to be an invention. More precisely, software often constitutes an element of an invention consisting of some

[36] KENNETH NICHOLS, INVENTING SOFTWARE: THE RISE OF "COMPUTER-RELATED" PATENTS, 3 (Quorum Books, Westport, 1998).

[37] IAN J. LLOYD, INFORMATION TECHNOLOGY LAW, 308 (Butterworths, London, Edinburgh, Dublin, 2000, 3rd ed.).

"functional interrelation between technical components of a system, *e.g.* the architecture of a processor and the particular way of processing data in such a processor."[38]

Because the border lines in such approach are blurred, controversies arising from allowing software patents are by far higher than those that have resulted from extending copyright protection over computer programs. Some vague legal guidelines for software patentability may already be found on an international level, within the framework of WTO in TRIPS Art. 27 providing for a general obligation to grant patents in all fields of technology, which "is largely taken to encompass software, when used to solve a technical problem"[39], partially in the light of Arts. 27(2) and 27(3) which do not mention software as capable of being excluded from patentability.[40] On the other hand; as Tripathi and others rightly point out, patenting things that do not meet general patentability requirements and are not inventions is not the objective of TRIPS, although Art. 1 allows to establish a more extensive protection than the one provided for by the agreement.[41]

In the U.S., which is one of the TRIPS signatories, as of 2000, there have been over 40,000 software patents and the number has been growing by few thousand every year.[42] For the time being, the ruling decision in the American debate on software patentability has been *State Street Bank & Trust v. Signature Financial* Services,[43] where the Court of Appeals for

[38] Yannis Skulikaris, *Software-Related Inventions and Business-Related Inventions; A review of practice and case law in U.S. and Europe*, Patent World, February 2001, at 26.

[39] *Id.* at 27 (2001).

[40] Daniele Schiuma, *TRIPS and Exclusion of Software "as Such" from Patentability*, No.1 Vol. 31 IIC International Review of Industrial Property and Copyright Law 36, 40 (2000).

[41] R C Tripathi *et al.*, *Patenting of Computer Software: Status and Approach*, Vol. 7 Journal of Intellectual Property Rights 128, 129 (2002), adding that in the early 90s, when TRIPS was drafted and discussed software was mostly considered not to constitute inventions but algorithms, at best just discovered (*Id.* at 130)

[42] Peter Toren, *Software and Business Methods are Patentable in the U.S. (Get over it)*, Patent World, September 2000, at 7.

[43] *State Street Bank & Trust v. Signature Financial Services*, 149 F.3d 1368 (Fed. Cir. 1998), cert. denied, 119 S.Ct. 851 (U.S. Jan 11, 1999).

the Federal Circuit was presented with a computerized algorithm for managing an investment fund structure and held that it constitutes a patentable subject matter,[44] which should be evaluated under the usual test of usefulness, novelty and non-obviousness.[45] According to Ogden, although the earlier case law could have led some to believe that physicality of an invention in the sense of transformation of a tangible article to a different state is the necessary condition of patentability, *State Street Bank* clarifies that it is not, at least not any longer.[46] The court; however, did set the limit – it is not the physicality, but the usefulness of an algorithm. Thus, "merely abstract ideas constituting disembodied concepts or truths that are not 'useful'"[47] are not patentable.

In Europe, the European Patent Convention of 1973 (EPC) excludes computer programs from the understanding of inventions in Art. 52(2)(c).[48] However, pursuant to Art. 52(3), patentability is excluded only to the extent to which an application relates to computer program "as such".[49] "Clearly, the interpretation of [*as such*] is a key issue in determining the patentability or otherwise of inventions in which the inventive step falls within a computer program."[50] It has been suggested that the *ratio legis* of Art. 52(3) was not to exclude patenting of computer programs absolutely and unconditionally.[51] In any case, the interpretation given by European Patent Office (EPO) seems not to be too restrictive; as of now (2004) already 30,000 software patents have been granted by EPO and the number has

[44] Christopher L. Ogden, *Patentability of Algorithms After State Street Bank: The Death of the Physicality Requirement*, No. 10 Vol. 82 Journal of Patent and Trademark Office Society 721, 722 (2000).
[45] Toren, FN 42 at 8.
[46] Ogden, FN 44 at 724 *et seq*.
[47] 149 F.3d 1368, 1373.
[48] European Patent Organization is not a signatory of TRIPS, though most of its members are.
[49] In fact, there are doubts whether the "as such" exception is allowed by TRIPS. Certainly, no TRIPS obligations apply to EPO and there is no direct obligation for TRIPS signatories to bring EPC in line with it. See Schiuma, FN 40 at 45 and 50 (2000).
[50] Rowland & Macdonald, FN 21 at 68.
[51] Skulikaris, FN 38 at 27.

been growing by 3,000 every year.[52]

Relying on the case law of Appellate Body, EPO searches the applications containing claims directed at computer programs for technical effect when analyzing whether they constitute a patentable subject matter. This is because the exception of EPC Art. 52(3) has been generally understood as excluding non-technical items from the understanding of "invention". To pass the test for patentability, "technical effect" has to go beyond mere interaction between hardware and software.[53] According to the current Guidelines of EPO, such "further technical effect" may be found

> in the control of an industrial process or in processing data which represent physical entities or in the internal functioning of the computer itself or its interfaces under the influence of the program and could, for example, affect the efficiency or security of a process, the management of computer resources required or the rate of data transfer in a communication link.[54]

EPO crafted the term "computer-implemented invention", which encompasses not only methods of operating a machine or machines designed to perform a method.[55] In the late 90s this term was broadened in the Appellate Body decision T 1173/97[56] to include "a computer program claimed by itself" if only the technical effect was present. Certainly, such interpretation leaves the reader puzzled by trying to find differences between "program claimed by itself" and "program as such". Attridge answers that the practice of EPO Appellate Body goes into the direction of construing the "as such" exception as only presumption that computer programs do not have technical effect and thus allows to grant

[52] Foundation for Free Information Infrastructure, *Software Patents in Europe: A Short Overview*, available at: http://swpat.ffii.org/lisri/intro/index.en.html.
[53] Skulikaris, FN 38 at 28.
[54] Guidelines for Examination in the EPO, 2003, 45-46.
[55] *Id.* at 45.
[56] *International Business Machines, Corp./Computer program product*, Decision of Technical Board of Appeal 3.5.1 dated 1 July 1998, T 1173/97 (OJ 10/1999, 609).

patents for the "written text ... without the need of technical means."[57]

There is a heated current debate on software patents within the European Union in connection with the proposal for the Directive on the patentability of computer-implemented inventions (Software Patents Directive).[58] The proposal has already been amended and at the time of this writing it appears that much still may change, especially after the recent Enlargement. The directive is presented by its supporters as the response to the unclear practice of EPO just described and divergent approaches towards software patentability in the Member States. Briefly speaking, the directive would not allow to patent the computer program itself but the invention embodied in the program and the proposal provides some guidance how to distinguish between these two. Although the directive is more elaborate on this point than EPC Art. 52(3), its opponents consider it just an attempt to confirm the lenient practice of EPO and the way to make computer programs *de facto* patentable. It should be kept in mind that even if the directive is adopted, it will not create any directly effective law; thus, EPC and national laws will remain the source of relevant rules.

Using a formal legal language, but not with regard to any particular jurisdiction, invention is either a product or a process, which satisfies all the following: (1) it is new; (2) it involves an inventive step; (3) it is capable of industrial exploitation; and (4) is not excluded from patentable subject matter.[59] These must be analyzed by authorities in every patent application. Certainly, there are some specific questions arising only in connection with software, such as whether it produces the "technical effect", but there are no reasons to believe that there would be any relating to Open Source Software only. Moreover, because

[57] Daniel J. M. Attridge, *Challenging Claims! Patenting Computer Programs in Europe and the USA*, 1 INTELLECTUAL PROPERTY QUARTERLY 22, 44 (2001).
[58] 6580/02 PI 10 CODEC 242.
[59] LLOYD, FN 37 at 314. See also *e.g.* European Patent Convention Art. 52-57.

Open Source Movement does not use patents to protect their software, the questions of patentability relevant for the purpose of this paper arise rather on a policy level as a part of discussion on proper system of Information Technology legal protection.

1.4 Trade Secrets Protection of Computer Programs

Another way to protect computer programs is trade secrets law.[60] Taking U.S. law as the point of reference, the protected subject matter of trade secrets is similar to that of patents – it can be methods or ideas. An important advantage; however, is that trade secrets do not require compliance with such burdensome patent law requirements as novelty and non-obviousness. The protection, though, is quite weak as it does not provide the holder with any monopoly, does not prevent independent creations and may be lost if the information becomes public.[61] The very idea on which trade secrets protection is based is confidentiality. It is the exact opposite of Open Source Software ideology and it may be said that trade secrets clearly do not protect Open Source Software.

1.5 Contract Law and Licensing of Computer Programs

The overwhelming majority of contracts in the trade of computer programs are drafted in the form of a copyright license. It may seem unusual, at least if compared to the trade in other copyrighted works that after fixation are distributed as any other tangibles, with the predominant use of the sales contract. The reasons for such difference will be discussed in the subsequent section as well as in other parts of the paper. It suffices to say here, that

[60] "Trade secrets" is an American term. In the UK, the analogous set of rules is usually referred to as "the law on breach of confidentiality". In various continental systems "confidential information" or "knowhow" is often used. TRIPS regulates this matter in Sec. 7, Art. 39 titled "Protection of Undisclosed Information". It must be kept in mind that trade secrets are not usually regarded as part of intellectual property laws but rather regulated by general civil law provisions or sometimes in the laws prohibiting unfair competition.

[61] Carey R. Ramos, David S. Berlin, *Three Ways to Protect Computer Coftware*, 16 No. 1 COMPUTER LAWYER 16 (1999); Victoria A. Cundiff, *Protecting Computer Software as a Trade Secret*, in: 507 PRACTISING LAW INSTITUTE, 18TH ANNUAL INSTITUTE ON COMPUTER LAW 761 (1998); Alois Valerian Gross, *What is Computer "Trade Secret" under State Law*, 53 AMERICAN LAW REPORTS 4TH 1046.

software licenses play an important role in the rapid growth of this trade and allow distributors to make their products easily and quickly available to the biggest number of customers.

Putting aside copyright law considerations and looking from the contract law standpoint software licenses can be described as contracts of adhesion, with the terms fixed by the seller in a standard form. Mass-market software licenses usually come in the form of "shrink-wrap" or "click-wrap". The first name refers to the method of distributing the software in a box shrink-wrapped in cellophane. In the early times of mass-market software industry license terms used to be printed on the box. Nowadays, they are put inside the box together with the program on a medium and printed manual. On the box, there is a reference to the terms and a notice stating that the removal of the cover, use of software or failure to return it after some fixed period constitutes assent to the terms of license.

In the "click-wrap" case, which is far more popular nowadays, the terms are presented on the computer screen, during the interactive process of installation or before that, at the time the program is downloaded from the Internet, either of which is carried on only upon clicking on the "I agree" button. Sometimes it is possible to click on the button only after scrolling down through the whole license or there may be other slight differences.

There are many other names in use for similar contracts, not necessarily constituting a copyright license or having software as its subject matter, because the criterion here is the procedure used in the contract formation. For example, "browse-wrap" refers to contracts concluded by surfing on a particular web page. The user is usually only informed that the on-line service is subject to specific terms, but not required to indicate whether he has read them or not. In other cases there may be only a link to "terms of use" at the bottom of the page providing the actual service.

To the extent that software licenses come in the form of a "-wrap" contract, the discussion on their legal consequences forms a part of a wider dispute concerning the use of standard-form contracting with the help of new technologies. Although the procedure used for contracting may seem extremely novel and rapidly changing, the legal debate is focused on such traditional concepts as "offer", "acceptance" *etc.* To the extent this discussion touches matters specific for Open Source contracting, it will be presented in this paper.

1.6 Theory and Practice of Current System of Legal Protection

Software legal protection rules are a part of the whole system of regulating intellectual property. It is believed that the rationale and ideology underlying intellectual property laws are different in the U.S. and continental Europe. American law is based on the "incentive theory" that sees in the protection for intellectual property an incentive for creative and innovative activity.[62] The U.S. Constitution gives the Congress the power

> [t]o 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.[63]

The monopoly awarded for authors, inventors and other intellectual or industrial property rights holders is believed to serve as such an incentive and promote the Progress.

Continental laws differ from American, or broadly speaking, common law approach, as they shift the importance from providing the incentive to increase the public good towards the protection of authors and their property rights in the creation. This "rights-based" theory can be seen in such legal documents as the Berne Convention or numerous national laws. The

[62] Drexl, FN 13 at 1.
[63] U.S. Const. Art. I Sec. 8 cl. 8.

result is not only the different terminology (“copyright” in common law versus “author's right” in Continental laws) but what is more important the existence of specific legal institutes, such as moral rights and other substantial differences. Logically, after extending the intellectual property legal protection over computer programs these respective theories are intended to apply to them.

The goal of intellectual property laws system, especially when considered from the American incentive-based standpoint, is to balance public and private interests. The society has a clear interest to secure for itself a constant supply of innovation (“the Progress”). The innovation constitutes of knowledge and its expressions or applications. Knowledge is a public good in economic sense, which means that it is capable of being consumed by many people at the same time without decreasing its value (non-rivalrousness) and that it is extremely hard to differentiate between consumers, for example in order to exclude free-riders from consumption (non-excludability). Although the supply and consumption of knowledge lies clearly in the public interest, it is believed that no private interests in producing it would exist unless a legal system for protection was provided.

Intellectual property laws constitute one system designed to provide incentives for the production of knowledge (the others are for example, government procurement and subsidies). The balance that they are trying to strike is the one between proprietary “control” over the innovation and free public “access” to it. As Oddi rightly points out, the market regulated by this system is the one for knowledge, not knowledge-related goods; thus, it is extremely important to evaluate it not by taking under consideration the number of products and services it makes available for consumption but first and foremost the amount of knowledge it transfers to the public.[64]

As it was presented in the previous sections, software may benefit from at least three legal methods of protection – copyright, patent or trade secrets. Each of them has been drafted for a different subject matter and their usefulness may vary depending on the kind of software or even on its particular element. For example, computer program "as such" should not be patentable under EPC, but it will be rather easily covered by copyrights. In any case, none of various intellectual property tools serve software distributors ideally. Trade secrets are too weak, patents difficult to obtain and copyrights limited to the protection of expressions not ideas. Not only copyright law grants third parties the access to programs' underlying ideas, but also it allows to a great extent the copying of programs' main source of value – their behavior. This is mainly due to a weak protection of non-literal elements and no prohibition of independent creations. Thus, together with visible efforts to make patent protection more readily available, software vendors reached for private contract long ago, which, especially in business-to-consumer trade, allows to extend much power over the buyers. Precisely speaking, software licenses are used to make the users contract away their rights provided for in copyright statutes and prevent the application of such institutes as fair use or first sale, to the extent it is legally possible.

It is usually raised that the main reason for the use of software licenses instead of simple sales contracts stems precisely from the specificity of software. Contrary to books, paintings *etc.*, every time computer program is used it has to be copied (from some storage device such as hard disk to the computer's RAM and then, command by command to the CPU), copying being the very action allowed only for the copyright holder. Without any,

[64] A. Samuel Oddi, *An Uneasier Case for Copyright than for Patent Protection of Computer Programs*, 72 NEBRASKA LAW REVIEW 351, 368 (1993).

express or implied, permission every user would become copyright infringer. Certainly this fact alone is not decisive, as with the help of such well-established institutes as fair use or implied licenses users could use programs legally purchased. Moreover, many copyright legislators long ago amended statutes in order to allow users all these necessary actions.

The prevailing reason why licensing, not selling, software is so popular is the fact that software distributors are at pains to protect themselves from at least two kinds of free-riding. First of all they want to protect their consumer base and secure incomes, endangered by piracy. Secondly, they aim against competitors, who would like to free-ride on developers R&D investment and use ideas available after decompilation and other study of programs. The first obstacle for free-riders results from technical means making it difficult to copy the program without the use of sophisticated tools and the practice of distributing programs in object code only, which does not allow for an easy access to ideas and knowledge embodied in the program. Software licenses form a second level of security measures – they contain express clauses prohibiting decompilation, copying or even using the program on more than one computer. Thus technical means are accompanied with the threat of legal remedies.

Standard software licenses usually contain liability limitations and warranty disclaimers, excluding developers and vendors liability to the extent permissible by applicable law. Such clauses are just examples of the use of contract law flexibility and obviously could be inserted in any other type of contracts; thus, their existence is not the reason why the form of a copyright license is the most popular. They are the effect of one more special feature of software, which has already been mentioned – the existence of software bugs. If developers were to release software containing no mistakes, they would be strongly discouraged from undertaking any development at all. Moreover, if they were to be held responsible for even a fraction of damages resulting from the use of buggy software it would definitely make their business very risky and unprofitable, unless they could economically insure themselves from such liability by raising prices and shifting the burden towards users. Liability limitation clauses are thus an alternative for such price shift and one more example how developers' interests are protected with the use of contract law.

Apart from this clear benefit for software distributors, mass-market software licenses allow them to take advantage of previously described “network effects”. Because of their adhesive character they result in quick and easy conclusion of an enormous number of contracts allowing to increase the value of a particular product in the eyes of consumers and gain the edge over competition.

To sum up, the practice of the current system of software legal protection is very much developers- and distributors- oriented. Despite the basic theoretical goal of intellectual property laws to balance the control and access, the resulting effect for the users and general public is practically no access, especially considering the fact that legal tools are backed up with technical anti-circumvention measures. Users play the passive role of the consumers of the end product and are at the mercy of developers in case they receive buggy software or if they expect some additional functionality from the product they have already purchased. This is the trade-off of the current system. Certainly, similar consumers' role in the market of traditional tangible goods is understandable, but such products could not be easily adjusted to their needs, as would be the case with software if only there were no legal and technical measures used to prohibit that.

The use of law as described above constitute the system of protection of so-called “proprietary software”. This paper is devoted to describe and analyze a very different phenomenon – Open Source Software. But before precisely defining the subject matter, discussing its elements and other details it is necessary to introduce some historical and other basic information about Open Source Movement.

Chapter 2 - History and Present State of Open Source Software

Open Source must be perceived as a process, still developing and having rich and quite complicated history. This history will be briefly presented here in order to provide the reader with the necessary background.

The signs of Open Source may be found in the earliest days of computer and software industry. In fact, one may argue that it was the starting point from which programs evolved into proprietary and copyrighted form. In these historical times, somewhere around the middle of the last century, computers were scarce, unbelievably big and impossible to operate by a layman. Computer programs were often written only for one machine as computers lacked compatibility. Even for some time after this had changed software was perceived mainly as a part of hardware not as a marketable product. Because no real software market existed, there were no practical issues of computer programs being copyrighted, licensed or pirated.

2.1 UNIX

The situation started to change with the development of the first portable operating system in the 70s – UNIX. The development of the system had been initiated by Bell Labs researchers and was initially designed for the computers used by AT&T. Its portability meant it could be implemented on various machines and made it probably the first piece of software that could be sold to users of many different computers. However, because of antitrust restrictions on software marketing, the company decided to share UNIX freely. It was allowing the licensees to access the source code in an attempt to receive their feedback.[65]

[65] Chris DiBona, Sam Ockman, and Mark Stone, eds., Open Sources: Voices from the Open Source Revolution (O'Reilly, 1999); Lawrence Lessig, The Future of Ideas: the fate of the commons in a connected world , (Random House, New York, 2001).

The University of California at Berkeley was one of the institutions that took the system for the use, analysis and further development. Numerous programmers, university professors and students contributed improvements and extensions to the system. In 1977 Berkeley released its first Berkeley Software Distribution (BSD), consisting of the system together with various application programs. For some time AT&T and Berkeley exchanged innovations and parts of programming code; however, without joining their distributions into a single product. In 1984 AT&T was broken up and no longer remained subject to restrictions on software marketing; it decided to start charging for licensing their UNIX released under the name System V. Licenses giving access to source code were still issued, but were expensive and included non-disclosure clauses. As BSD used much of AT&T's code, all its recipients had to obtain a license from AT&T too. Only in 1989 did BSD Networking Release 1 appear, from which all AT&T-owned code was retracted and substituted by programs written solely by the University.

In the 90s, BSD continued being marketed by Berkeley Software Design, Inc. (BSDI), a company linked to the University. Activities of BSDI gave rise to concern of AT&T's subsidiary, Unix System Laboratories (USL) the assignee of the rights to UNIX, which filed a lawsuit against BSDI together with The Regents of the University of California in 1992.[66] USL initially pleaded eleven claims for relief, including federal copyright and trademark claims, together with state contract, tort, and trademark dilution claims. The central issue of the case was whether BSDI and Regents appropriated parts of USL's UNIX and then used and distributed these parts without authorization in violation of plaintiff's copyrights and

[66] 1993 Copr.L.Dec. P 27,075, (27 U.S.P.Q.2d 1721); 27 U.S.P.Q.2d 1721; 1993 Copr.L.Dec. P 27,166, (86 Ed. Law Rep. 738, 29 U.S.P.Q.2d 1561).

trade secrets.[67]

More precisely, the case concerned one of the AT&T's UNIX versions called V32, which was licensed to the University with permission to create derivatives and, to the extent that the derivatives were free of proprietary information, to distribute them without restriction.[68] The University exercised its contractual right by creating its BSD releases and distributing them. According to USL, the BSD Net2 release was an infringement of the license since it contained AT&T's proprietary code. Indeed, USL's hired expert found portions of 32V code included in the Net2 system.[69] The court agreed with the parties that the size of overlaps (which was merely 130 lines out of 1.3 million in some part) is insignificant and analyzed their nature. There were three kinds of them: (1) overlaps in variable parameter and function names; (2) the text of noncoding comments; and (3) the actual sequence of the instruction code itself.[70] According to the court, USL had failed to demonstrate likelihood that it can successfully defend its copyright in 32V[71] or likelihood to succeed on the merits of its claim for misappropriation of trade secrets;[72] therefore, the preliminary injunction prohibiting BSDI from further distribution of BSD was not granted.

The case was not decided on its merits, as the parties started settlement negotiations, which continued until 1994, when USL with all its rights to AT&T's UNIX had been already acquired by Novell. Although the exact terms of the settlement remain confidential, the crucial parts were summarized by McKusik: the parties agreed to remove three out of 18,000 files from the Net2 Release and USL stipulated not to sue anyone using the 4.4BSD-Lite

[67] 1993 Copr.L.Dec. P 27,075 at 1.
[68] *Id.* at 2.
[69] *Id.* at 3.
[70] *Id.*
[71] *Id.* at 15.
[72] *Id.* at 18.

release that came to being in this way.[73]

Not only all the questions of important material facts were not decided during that historical lawsuit, but also no issues of law concerning, for instance, validity of BSD license were considered by the court. However, the facts outlined above and the dispute over them has to be taken under consideration when trying to untangle current Open Source legal problems, namely the *SCO v. IBM* dispute which is shortly addressed later in this paper. In any case, *USL v. BSDI* is an important event in the development of one of the major pieces of Open Source Software – three flavors of BSD system. These independent Open Source, UNIX-like operating systems came to being after 1992 on the basis of one of Berkeley distributions designed for 386 PC architecture, 386/BSD. Its users formed groups for the purpose of cooperating in debugging and enhancing the system under the names of NetBSD and FreeBSD. Additional group, OpenBSD, emerged in the mid-90s. Despite the common starting point the programs they distribute constitute different operating systems. After the settlement of *USL v. BSDI* case, all groups incorporated 4.4 BSD release to their distributions. From then on, they are maintained and developed solely by their users, though the participation of anyone is welcomed.

2.2 Free Software and Project GNU

Open Source Movement could not have come into being without the Free Software Movement. It has been initiated and the term "Free Software" has been introduced by Richard M. Stallman, who is still the unquestioned leader of the movement. During his work at the Massachusetts Institute of Technology, Artificial Intelligence Laboratories (MIT Labs) from the early 70s, Stallman witnessed that the practice of sharing and exchanging software just as

[73] Marshall Kirk McKusick, *Twenty Years of Berkeley Unix*, in DiBona *et. al.*, FN 65 at 27, 34.

"cooking recipes" was going into the past.[74] At that time, as Lessig describes it:

> [T]he openness of commercial code [*i.e.* software] began to change. As products became more numerous and users became more diverse, and as the cross platform compatibility of programs grew, the companies producing these products exercised more and more control over how the products might be used. ... Users became less partners in the process of developing and using computer systems and more consumers. And suppliers of code were less eager to permit their code to be copied by others.[75]

From the beginning of the 80s, programmers were required to sign non-disclosure agreements for the software. According to popular anecdote, Stallman originated Free Software Movement because he was refused access to the control program of a new printer that used to be freely accessible before and thus subject to many useful modifications. Proprietarizing computer programs and making their source code secret meant for him the end of a friendly cooperation era. Instead of joining the main stream he left his job at MIT Labs and started developing the whole system of computer programs that could be used, modified and redistributed freely under the name Project GNU.[76] Supervision and maintenance of Project GNU is the main purpose of the Free Software Foundation (FSF), established by Stallman and his followers in 1985.

2.3 Linux

"Linux" is the name used for the most popular Open Source operating system.

[74] Richard M. Stallman, *The GNU Operating System and the Free Software Movement*, in DiBona *et al.*, FN 65 at 40, 40.

[75] Lessig, FN 65 at 52.

[76] GNU stands for "GNU is not UNIX", *i.e.* not a proprietary, closed-code system. (Richard M. Stallman, *The GNU Project*, in Stallman, FN 92 at 17) However, Stallman decided "to make [GNU] compatible with UNIX, so that it could be portable, and so that UNIX users could easily switch to it." (Stallman, FN 74 at 41). Instead of explaining why GNU actually IS UNIX, it suffices to say here that the name "GNU" is just one of many plays on words beloved in the programming community.

Strictly speaking; however, it is only the name of kernel program[77] written in 1991 by Linus Torvalds, then a student at University of Helsinki, Finland. At that time, GNU Project, although much advanced, was still lacking a satisfactorily working kernel, which would allow for a release of the whole GNU operating system. Because Torvalds decided to share his program with the world, the kernel was successfully combined with the rest of GNU programs thus forming a complete operating system, which FSF recommends calling "GNU/Linux". Linux, the kernel, is constantly developed by Torvalds and programmers more or less closely cooperating with him.

Torvald's decision to make the kernel freely available, together with a similar step taken by intellectual property rights holders in other elements of GNU/Linux system, made it possible for everyone to create his own version of the complete operating system by tuning up the kernel and merging it with any other necessary programs. Because the kernel is a crucial part, the operating system based on Linux is commonly referred to using its name. Linux or GNU/Linux, the system, is structured into distributions similarly to BSD and its "cousins" (FreeBSD, NetBSD and OpenBSD). Distributions consist of operating system and various sets of application programs adjusted to work together. Many of these programs come from the GNU Project; however, there are plenty of applications written for the Linux system by other developers, including those working under the proprietary model.

There are many different GNU/Linux distributions prepared by persons who track the Internet for the newest versions of all the programs or develop some of them themselves. The idea of Open Source intends to allow this to be done by everyone, both in technical and legal sense. Such people sometimes cooperate under informal structures but often form

[77] Kernel is the core program of every operating system. It is responsible, generally speaking, for the communication between other programs and the computer.

entities. Many of them work on non-profit basis, examples of which are Debian or Slackware. There are also plenty of commercial companies (*e.g.* Red Hat, SuSE, Mandrake) that have established successful businesses in selling GNU/Linux distributions prepared by them. Therefore, no single operating system called "Linux" exists, as there is no single car brand.[78] In order to be formally precise, each time it should be indicated which particular Linux distribution the speaker has in mind (*e.g.* Debian GNU/Linux, SuSE Linux, *etc.*). However, it matters mostly from a technical standpoint and legally speaking there are no significant differences between distributions that would be overseen by not making such indication.

2.4 Open Source Initiative

In the 80s and also some time after Torvalds' contribution allowed for putting together complete and working operating system, Open Source programs were not known to the wider public. Even after one of the first commercial companies marketing Open Source, Cygnus Solutions was founded in 1989, these programs continued to be written by the hackers and for the hackers.[79] It required quite sophisticated programming knowledge and skills to make Open Source programs running on any computer. The other reason why Open Source seemed not to be a good idea of doing business was what Eric S. Raymond called "'free-speech/free-beer' ambiguity,"[80] which will be thoroughly described in the section comparing Free Software and Open Source Movements. It took some time for the developers to make Open Source programs user-friendly and for the firms to get accustomed to the idea

78 This comparison was made in ROBERT YOUNG & WENDY GOLDMAN ROHM, UNDER THE RADAR: HOW RED HAT CHANGED THE SOFTWARE BUSINESS – AND TOOK MICROSOFT BY SURPRISE (Coriolis Group Books, Scottsdale, Arizona 1999).

79 In the words of Stallman, "hacker" means "someone who loves to program and enjoys being clever about it." (Stallman, FN 74 at 48). In this paper it will be used to indicate a highly trained professional programmer, definitely much above the level of an average computer user. In any case "hacker" should not be mistaken for a "cracker", a person using Information Technology in order to commit various crimes.

80 Eric S. Raymond, *The Revenge of the Hackers*, in DIBONA *ET. AL.* FN 65 at 118, 120.

of selling something, which is accessible for everybody for free anyway.

Open Source Initiative (OSI) was established by Raymond and others in order to introduce and promote this innovative approach to the world of business, predominantly accustomed to the proprietary model. One of the first moves was the change in terminology – from “Free Software” to “Open Source”, which was intended to help to avoid explaining what is really important in Open Source from the economic point of view. Secondly, OSI started to issue Open Source Certificates in order to establish a market brand and promote standardization in Open Source licensing.

2.5 Various Open Source Projects

Open Source Software constitutes not only of operating systems such as GNU/Linux or three BSD “cousins”. It is also a plethora of application programs serving various purposes, from solving sophisticated technical tasks to doing what the majority of common users usually expect – office applications and game software. Open Source projects presented in this paper were chosen using two criteria. Firstly, they are one of the most popular pieces of Open Source Software. Secondly, attention was given to the legal issues triggered by them. Thus, care was taken to choose for discussion projects, the legal protection of which is the most important and at the same time interesting. Certainly, because of the big number of them, the choice presented here clearly does not constitute of all projects that are worth analyzing. On the other hand, many legal institutes are so popular that this choice may be considered sufficient for providing enough examples. Thus, apart from the Linux kernel, some of the most popular distributions of GNU/Linux system referred to at various occasions and three BSD operating systems, the following will also be discussed: Apache HTTP Server, Mozilla Web Browser and OpenOffice.org Office Suite.

2.5.1 Apache HTTP Server

The Apache Project aims at creating "a robust, commercial-grade, featureful, and freely-available source code implementation of an HTTP (Web) server."[81] The development of the Apache Server was started in 1995 by a group of webmasters in the need of a server for their work. Apache is based on a program called httpd, developed at the National Center for Supercomputing Applications (NCSA). Httpd was released into public domain and many webmasters who developed their own extensions and bug fixes wanted to exchange them and join their efforts for further development of the server.[82] They established a forum known as "the Apache Group" and quickly started releasing the program that they developed collaboratively with many volunteers from all over the World.

In 1999, members of the Apache Group formed the Apache Software Foundation to support the development of the Server and other related projects.[83] All project participants are supervised by the Apache Group under the rule of "meritocracy – the more work you have done, the more you are allowed to do".[84] They employ quite formal rules *e.g.* concerning voting on the inclusion of proposed improvements into subsequent versions of the Server.

The Apache Server remains free and the Apache Group states:

> To the extent that the protocols of the World Wide Web remain "unowned" by a single company, the Web will remain a level playing field for companies large and small. Thus, "ownership" of the protocol must be prevented, and the existence of a robust reference implementation of the protocol, available absolutely for free to all companies, is a tremendously good thing.[85]

[81] The Apache Software Foundation, *About the Apache HTTP Server Project*, available at: http://httpd.apache.org/ABOUT_APACHE.html.
[82] *Id.*
[83] The Apache Software Foundation, *Home Page*, available at: http://www.apache.org/
[84] The Apache Software Foundation, FN 81. See also: The Apache Software Foundation, *How the ASF works*, available at: http://www.apache.org/foundation/how-it-works.html#meritocracy.
[85] The Apache Software Foundation, FN 81.

Additionally, the free availability of Apache is considered a prerequisite for its users to contribute back to the project with their improvements and bug fixes.

2.5.2 Mozilla Web Browser

The Mozilla Project came into being because of a commercial company's deliberate decision to choose Open Source as their model of business. In January 1998, Netscape Communications announced it would start distributing their Netscape Communicator Standard Edition 5.0 web browser and some other applications for free and with access to source codes.[86] Later on, the company founded Mozilla Organization, a forum for all contributors involved in the project, which main aim is to facilitate reaching consensus about the project's development.

Today, the project encompasses a number of applications, with the Mozilla Web Browser and more recently, Mozilla Firefox Web Browser, being its flagship. The project is structured into modules supervised by "module owners" chosen because of their merits for the project, who decide about modifications and improvements to be submitted for the inclusion in subsequent releases of Mozilla products.[87] The process is intended to be a self-regulating "meritocracy", because "module owners" can be replaced by persons whose code proves to be better. In fact, the whole Mozilla project can be replaced thanks to the openness of its structure, as users might simply turn to those distributors who supply better code if this code for some reason would not be approved by the Mozilla Organization.

Netscape still develops and distributes products such as Netscape Communicator and

[86] Netscape, *Netscape Launches Aggressive 'Unlimited Distribution' Program For New Free Client Software*, available at: http://wp.netscape.com/newsref/pr/newsrelease560.html; Netscape, *Netscape Announces Plans to Make Next-Generation Communicator Source Code*, available at: http://wp.netscape.com/newsref/pr/newsrelease558.html.

[87] The Mozilla Organization, *Mozilla Roles and Responsibilities*, available at: http://www.mozilla.org/about/roles.html.

there is a both-ways exchange between them and Mozilla, as the code is shared between the applications.

2.5.3 OpenOffice.org Office Suite

OpenOffice.org is a set of office applications, which includes a word processor, spreadsheet, presentation manager and drawing program.[88] Its appearance (user interface) and available set of functions are similar to other office software and it also allows exchange of documents with other applications. There are versions of OpenOffice.org running on Linux and other UNIX-like operating systems, but it can also be used with Microsoft Windows.[89]

OpenOffice.org is developed by Sun Microsystems with the help of contributors from all over the World. At the same time Sun maintains the commercial version of the application – StarOffice, with which OpenOffice.org has similar link to the one between Mozilla and Netscape. Contributors willing to participate in the development of OpenOffice.org are required to register, subscribe to a mailing list and join one or more sub-projects, which are devoted to particular tasks.[90] Project leads have voting rights concerning future development of the whole application.

---

[88] Sun Microsystems, *OpenOffice.org 1.1 Product Description*, available at: http://www.openoffice.org/product/

[89] Sun Microsystems, *download: Download Central*, http://download.openoffice.org/1.1.2/index.html.

[90] See generally: Sun Microsystems, *Contributing to OpenOffice.org*, available at: http://www.openoffice.org/contributing.html.

Chapter 3 - Role of Law in Development of Open Source Software

It has been indicated at the end of Chapter 1 that Open Source Software differs significantly from proprietary software. Hopefully, the brief overview of the movement's history and short descriptions of some of its major projects helped to present the technical difference. The purpose of current Chapter is to find out what the legal differences are and what the exact role that law plays in the development of Open Source Software is. In order to resolve these issues, it will be explored what parties are involved, what their interests are, which ideology they refer to, and finally, what use of law they make or intend to make.

3.1 Free Software v. Open Source

Generally speaking, both "Open Source" and "Free Software" refer to computer programs, over which users have total control. They can freely use the program for any purpose, modify and redistribute it. In order to be able to exercise these rights in practice, they are also given access to programs' source codes. Only then users are able to see how programs operate, find causes of errors, correct them on their own and even develop the programs by adding improvements. This is what distinguishes both Open Source and Free Software programs from "proprietary programs", which are distributed only in object code, or "binary" format. As it was already presented, the users of proprietary programs are prohibited by licenses to modify and redistribute them, and the allowed scope of use is significantly limited. Further distinction has to be made between Open Source, Free Software and public domain. Open Source and Free Software licensors do not waive copyrights, which is the case with public domain software.

Although "Free Software" seems to address the same phenomenon as "Open Source" there are some formal and historical differences. Stallman, who is the author of the term

“Free Software”, believes that access to source codes and ability to modify and improve programs are the natural right of users. He has developed this idea in many publications transmitting a strong ideological and ethical message referring to ideals such as freedom, free speech[91] and friendly cooperation. For example, a big part of Stallman's *GNU Manifesto* entitled *Why All Computer Users Will Benefit*,[92] draws a Utopian view of an ideal computer users community. According to Stallman “[o]nce GNU is written, everyone will be able to obtain good system software free, just like air.”[93] In another essay titled *Why Software Should Not Have Owners*[94] he argues against the copyright protection for computer programs comparing this profit-driven system of control with practices of mass censorship in the former Soviet Union. Being against the practice of the current system of intellectual property protection, Stallman nevertheless supports the underlying idea of this system – to supply the public with knowledge and innovation.[95] But above all, he puts the ideals of open and free cooperation, friendly sharing of knowledge and helping the neighbor.[96]

Stallman introduced the following Free Software Definition (FSD):

- Freedom 0: The freedom to run the program, for any purpose.
- Freedom 1: The freedom to study how the program works, and adapt it to your needs. (Access to the source code is a precondition for this.)
- Freedom 2: The freedom to redistribute copies, so you can help your neighbor.
- Freedom 3: The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. (Access to the source code is a

[91] But not free beer, which is stressed by almost everybody who writes on this topic. Distinction between free speech and free beer is made to show that freedom of access to source code does not mean it will always be given away gratis.
[92] Richard M. Stallman, *GNU Manifesto*, in: RICHARD M. STALLMAN, FREE SOFTWARE, FREE SOCIETY: SELECTED ESSAYS OF RICHARD M. STALLMAN 31, 34 (GNU Press, Boston, 2002).
[93] *Id.*
[94] Richard M. Stallman, *Why Software Should Not Have Owners*, in: STALLMAN, FN 92 at 45.
[95] *Id.* 46-48; Richard M. Stallman, *Misinterpreting Copyright – A Series of Errors*, in: STALLMAN, FN 92 at 77.
[96] Stallman, FN 94 at 49.

precondition for this.)[97]

Additionally, Stallman uses the term “copyleft”,

> [which] central idea ... is that [everyone is given] permission to run the program, copy the program, modify the program, and distribute modified versions – but not permission to add restrictions of their own. Thus, the crucial freedoms that define “free software” ... become inalienable rights.[98]

In other words, “copyleft” is the mirror image of “copyright”, as instead of allowing for proprietarization and privatization of software it helps to keep it free in the meaning of FSD. Both FSD and copyleft are expressed in the legal language of license clauses. Free Software Foundation (FSF) maintains a whole list of software licenses categorized as Free Software.[99] Thus, in the understanding given to it by FSF, a program is Free Software if only it complies with all four freedoms defined in FSD. It does not need to be “copylefted” but FSF encourages it in order to protect “the program's freedom”.

“Open Source” was introduced at the time of Netscape's announcement to open the code of their web browser by a group consisting of people such as Raymond or Bruce Perens, who subsequently formed Open Source Initiative (OSI).[100] They proposed a new term for an old phenomenon in order to denote the practical significance of access to source codes and to bring the idea under attention of the world of business. Although Stallman does not object to the commercialization of software and often explains that it had never been his intent that Free Software be available at no charge, it may be easily seen from his writings briefly summarized above, that the development of a successful Free Software business is not his

[97] Richard M. Stallman, *Free Software Definition*, in: Stallman, FN 92 at 41.
[98] Stallman, *The GNU Operating System and the Free Software Movement*, in DiBona *et al.,* FN 65 at 43.
[99] Free Software Foundation, *Various Licenses and Comments about Them*, available at: http://www.fsf.org/licenses/license-list.html.
[100] Open Source Initiative, *History of the OSI*, available at: http://www.opensource.org/docs/history.php.

main concern. Yet, in economic terms, the friendly sharing Stallman speaks about, amounts to reduced costs of tracing programming mistakes and gathering information about users' expectations. Thus, Open Source Initiative describes itself as "a marketing program for free software", based on "solid pragmatic grounds".[101] These are developed in three *Cases for Open Source*: *For Business*,[102] *For Customers*[103] and *For Hackers*,[104] presenting such practical benefits of Open Source as reliability and quality or independence from a single supplier.

According to the people behind OSI, "Free Software" is too ambiguous to attract commercial companies, mainly because it suggests that the programs are given away free of charge or are in public domain. "Free Software", especially in the light of Stallman's writings, is also strongly associated with "hostility to intellectual property rights, communism, and other ideas hardly likely to endear themselves to an MIS manager."[105] Whether it is the right association remains to be seen but in the minds or "Open Source" proponents the completely new term is supposed to clarify this misunderstanding and avoid entering a heated ideological debate. Additionally, Open Source Initiative distances itself from ideological and moral dilemmas, limits to economic arguments and "does not have a position on whether ideas can be owned, whether patents are good or bad, or any of the related controversies."[106]

Legal significance of "Open Source" should be mostly attributed to the Open Source Definition (OSD), which current Version 1.9 states: "Open source doesn't just mean access to

[101] Open Source Initiative, *Frequently Asked Questions*, available at: http://www.opensource.org/advocacy/faq.php.
[102] Open Source Initiative, *Open Source Case for Business*, available at: http://www.opensource.org/advocacy/case_for_business.php.
[103] Open Source Initiative, *The Open Source Case for Customers*, available at: http://www.opensource.org/advocacy/case_for_customers.php.
[104] Open Source Initiative, *The Open Source Case for Hackers*, available at: http://www.opensource.org/advocacy/case_for_hackers.php.
[105] Eric S. Raymond, *The Revenge of the Hackers*, in DiBona *et. al.*, FN 65 at 118, 120.
[106] Open Source Initiative, FN 101.

the source code. The distribution terms of open-source software must comply with the following criteria" and goes on to lay down 10 licensing conditions:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. [Detailed means of source code availability follow]

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

[Allowed licensing conditions of redistributing modified software]

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.[107]

Similarly to FSF, OSI scrutinizes software licenses and evaluates them as complying with OSD or not. It also issues certificates of compliance.

In order to determine and clarify the scope of this paper it is necessary to find what legal consequences the relationship between “Free Software” and “Open Source” has. It is possible to compare FSD with OSD but although they use normative language, they do not set rights and obligations. Certainly, from the legal point of view it is the software license that matters and since both organizations declare particular licenses conforming or not with their respective definitions it seems reasonable to compare the lists of these licenses and thus determine whether “Free Software” legally differs from “Open Source”. Such comparison reveals some differences, which for the purpose of this paper can be considered minor.[108] All the licenses discussed in this paper, namely GNU GPL, GNU LGPL, BSD License, Apache Licenses, MPL and SISSL are approved by both FSF and OSI as conforming with their respective definitions. Thus, if one limits the discussion to the legal aspects of these licenses

[107] Open Source Initiative, *The Open Source Definition,* available at: http://www.opensource.org/docs/definition.php.

[108] There are only two licenses approved by OSI, which are expressly not approved by FSF. There are many licenses included in one list only.

it does not seem to be a great formal mistake to use terms “Free Software” and “Open Source” interchangeably.

For Stallman though, “Free Software” is better than “Open Source” and he treats these two movements as substantially different in their values and ways of looking at the world, but still remaining “two political camps within the free software community.”[109] Although he recognizes proprietary software, not the Open Source Movement as “the enemy”, Stallman does not want to stop raising his ethical arguments and make any concessions of the users' freedoms. His main argument is that different words convey different meanings and “Open Source” was designed not to raise the point of users' freedom,[110] or maybe even in order to place “profit above freedom, above community, above principle.”[111] Simply speaking, Stallman was never against making profit from software, but strongly objects to every attempt at making it proprietary; he believes that “Open Source” blurs this idea.

Thus, even if legal differences are minor, there is an ideological gap between FSF and OSI. The former conducts a “moral crusade”, the latter merely a “marketing program”. It influences all legal questions to the extent that both exist within the same legal system of software protection. More precisely speaking, their rationales make them use the system to different ends. For example, GNU GPL drafted by FSF contains a very strong “copyleft” clause, which generally speaking is designed to legally force programmers to release their programs under GNU GPL if they are in a certain relationship with original programs covered by it. This so called “copyleft virus” is intended to spread the Free Software idea and make

[109] Richard M. Stallman, *Why “Free Software” is Better than “Open Source”*, in: STALLMAN, FN 92 at 55, 55.
[110] *Id.* at 59.
[111] Stallman, FN 98 at 48.

releasing proprietary software harder. In comparison, many commercial companies working under Open Source model use far weaker "copyleft" clauses or even "dual licensing schemes" which aim at satisfying both the hackers' community and those who follow proprietary tradition, thus not influencing anyone to adhere to any particular ideology. Furthermore, Free Software Movement is concerned with creating the whole system of Free Software and looks closely at the licenses whether they allow programs to be linked with each other. For this reason, FSF indicates that some of the licenses, although they make a program Free Software, are GPL-incompatible, which means that "a module released under such licenses cannot be legally combined with a module covered by GNU GPL in order to make one larger program."[112] OSI makes no differences between OSI-approved licenses and one reason for it is probably that commercial companies they address are not interested in cooperating at a scale, which Stallman would like to see between all computer users.

For the convenience of the reader and clarity of the text, the term "Open Source" will be used throughout this thesis, but by no means should it suggest that the Open Source Movement as a whole is homogeneous or ideologically neutral.

3.2 Hackers' Culture

In order to fully understand the Open Source phenomenon, it is not enough to distinguish between the ideologies of Free Software and Open Source. There are many different social groups and market players within the movement, all connected by the underlying idea, which is to share one's work and invite others to cooperate. It evolved from the academia, such as the one Stallman worked at before leaving MIT, which was based on open cooperation between scholars and students. This scientific community served as the

[112] Free Software Foundation, FN 99.

starting point for the hackers' culture, which further developed over the Internet to encompass many technologically-savvy people who like to solve challenging tasks together. Only subsequently, lured by the benefits OSI talks about, commercial companies joined them. Thus, all the participants in the Open Source phenomenon may be divided into non-profit and business players, and the division lines usually run along Free Software – Open Source distinction discussed in the preceding section. The criterion here is not the ideology but the predominant motivation to introduce or participate in Open Source projects. The purpose of this section is to present briefly the culture, which has evolved among non-profit Open Source participants and to evaluate its impact on the law.

An interesting example of the specificity of this culture is Debian Open Source project. It was started in 1993 by Ian Murdock, then a university undergraduate, in order to create a free GNU/Linux operating system and has evolved to be one of the most popular Linux distributions.[113] What is particularly interesting is that their distribution is developed under the rules of *Debian Social Contract*, a big part of which ("Debian Free Software Guidelines") served as a basis for Open Source Definition.[114] The current version of the Contract was "ratified" in April 2004 and contains rules of developing and distributing the system. It is hard to authoritatively determine the Contract's legal character as it is much more the organization's program statement and code of conduct than an agreement. At the same time it addresses the users and makes representations about the software contained in Debian distribution.

It is not the most important whether the clauses of the Contract could serve as a basis

[113] Debian Documentation Team, *A Brief History of Debian*, available at: http://www.debian.org/doc/manuals/project-history/.

[114] Software in the Public Interest, *Debian Social Contract*, available at: http://www.debian.org/social_contract.en.html.

for any legal action. The purpose for which it is mentioned in this paper is to show an example of rules governing the Open Source Community. Although sometimes they are spelled out in documents such as Debian Social Contract, many norms remain only moral or social and the hackers usually abide them regardless of their legal validity.[115] This has to be remembered when considering the legal aspects of more formal documents such as the Open Source licenses. Indeed, according to Gomulkiewicz, "the [GNU] GPL reads like a combination of a political tract, philosophy dissertation, how-to guide for non-lawyers, and [finally] lengthy, complex license contract."[116] The result is that many legal institutes have different meanings in the Open Source practice and they work not as a lawyer would expect them to work. Simply speaking, the hackers do not always behave in the way the legal understanding of their documents would suggest.[117]

Many other interesting social customs have been described and analyzed by Raymond in his essay *Homesteading the Noosphere*.[118] The main conclusion that can be drawn from this writing is that the hackers' cooperation is not so free and loose as it could be inferred at first glance from its friendly and predominantly not-for-profit character. The organizational hierarchy together with formalized cooperation procedures is quite well developed. For example, the black letter of Open Source licenses suggests that since everyone has permission to modify and redistribute, the development of particular project should

[115] For example, one of the first and still very popular Linux distribution, Slackware (originated in 1993 by a university student, Patrick Volkerding) does not have any constitution, philosophy *etc.* Their webpage (http://www.slackware.org) is more focused on technical issues.

[116] Robert W. Gomulkiewicz, *De-bugging Open Source Software Licensing*, 64 UNIVERSITY OF PITTSBURGH LAW REVIEW 75, 92 (2002).

[117] It is also interesting to see that the hackers treat the licenses they draft similarly to computer programs. They "upgrade" them and issue new versions, numbering of which resembles numbering of subsequent program releases. They also discuss "compatibility" of various licenses, whether they can be linked *etc*.

[118] Eric S. Raymond, *Homesteading the Noosphere*, available at: http://www.catb.org/~esr/writings/cathedral-bazaar/homesteading/

quickly “fork” into many competing products. Yet, in the long history of Open Source it happened only to BSD operating system if one considers the major projects. According to Raymond this is because “the open-source culture has an elaborate ... set of ownership customs.”[119] There is usually only one person or a small group considered to actually “own” the project, referred to as the “project coordinators”, who decide about further development of the project.

A quick glance at all the customs described by Raymond supports the conclusion that Open Source licenses backed up with social norms form a very flexible system aimed at preserving not the interest of a particular developer but the whole community. For example, in the proprietary world, frictions in certain company would stall the development of this company's projects, whereas when the cooperation in a particular Open Source group becomes too burdensome and social norms no longer can prevent their conflicts it is always legally possible for anyone to “fork” the project and continue working on it with other people. Thus, all the previous work is preserved for the community.

It seems that the main purpose of Open Source licenses is to protect the movement from the real “enemy” – proprietary software. As it may be seen from the German case described below, the main threat for an Open Source project is to be proprietarized and to have its source code closed. As all the advantages of Open Source come from the availability of the source code this would simply lead to the project's death or at least serve as a strong deterrent for participants to contribute. This is the point at which social norms are particularly strong and where there is a real threat of using the legal weapons of licensing clauses.[120] But

[119] *Id.* at 6.

[120] Moglen raises the argument that the acceptance of the license (GNU GPL) is only required when the user engages in any activity that is connected with the threat of proprietarization of Open Source Software. According to him, simple use, inspection or even experimental modification is not covered by GNU GPL. (Eben Moglen, *Free Software Matters: Enforcing the GPL, I*, 2, available at:

again, they do not serve particular developers as it is the case with proprietary licensing but the community and protect it from being deprived of the code. The most interesting aspect of it is, that this system is almost frictionless and works more efficiently than the one designed by proprietary software licensors. This conclusion may be drawn from the fact that so far there have been few litigations involving Open Source in comparison to many cases dealing with the enforceability of proprietary licenses. Eben Moglen, the legal counsel for Free Software Foundation, describes the procedures of enforcing GNU GPL and reports that it was never necessary to bring the matter before court and infringements were remedied quickly after notices or some negotiation.[121] According to him, the infringers have more incentives to comply with licenses and get the software for free, instead of stealing it.

The knowledge of Open Source community norms and license enforcement practice is also extremely important *de lege ferenda*. There is much attention given to Open Source Software by the European Union[122] and public administration of Member States,[123] which may lead them to draft changes in the system of intellectual property protection. In such case it must be precisely known what exactly to protect in Open Source Movement. Moreover, moral norms which seem to work fine already should not be ignored.

3.3 Open Source as Marketing Model

The hackers (the technologically-savvy individuals) voluntarily participate in Open Source projects mainly because they need the tools they develop (as is the case with the

http://moglen.law.columbia.edu/publications/lu-12.html).

[121] Eben Moglen, *Free Software Matters: Enforcing the GPL, II*, 1-3, available at: http://emoglen.law.columbia.edu/publications/lu-13.html.

[122] European Communities, *Europa – Information Society – Free & Open Source Software – Introduction*, available at: http://europa.eu.int/information_society/activities/opensource/index_en.htm

[123] See for example: werk21, *Bundestux – Pinguine ins Amt! [Bundestux – Penguins to the Office!]*, available at: http://www.bundestux.de/; Piotr Bolek, *Forum Rozwoju Wolnego Oprogramowania [Forum for the Free Software Development]*, available at: http://frwo.linux.org.pl/ (in Polish).

Apache Server), they want to help others in their free time or they want to gain recognition and authority. Obviously, the reason why commercial companies start or participate in Open Source projects is to profit. The indigenous system of Open Source collaboration allows all these players to realize their respective goals. This, however, is not done in the traditional way of the proprietary market, which clearly sets borders between firms – the suppliers and users – the consumers. The system of legal protection plays important role in making Open Source attractive for all the players.

It needs a bit of abstract and creative thinking to establish a successful business in the Open Source world. First of all, it is relatively hard to make an Open Source program a product and companies usually decide to opt for one of alternative "Indirect Sale-Value Models" described by Raymond.[124] But even so, especially in places of the World with poor Internet access some people are willing to pay for Open Source Software on tangible media despite it is anyway downloadable from a FTP server for free. They are often willing to pay more, if the software is preconfigured, tested and warranted. Thus, Open Source companies mainly focus on adding the value to otherwise free programs. Some other variant is to use Open Source Software to support the company's other products, which are designed for or work with it. Server manufacturers pursue this strategy in supporting the development of the Apache Server.

It is interesting, that these commercial goals are realized with the use of roughly the same system of legal protection which serves the hackers. Obviously, there are differences which cannot be reconciled and this is the main reason why many companies decide to draft their own standard form licenses rather than release their code under GNU GPL or BSD

[124] Eric S. Raymond, *The Magic Cauldron*, 13, available at: http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/.

License. One important difference is that the companies are usually not pursuing any world-wide crusade for the benefit of all users but modestly have in minds their shareholders only. For them, Open Source is just means, not the end itself and they usually want to balance between this and proprietary production models.

One popular idea is to make the company's products aim both markets at the same time and issue the same software under many licenses. The examples of such multiple licensing are Mozilla[125] and OpenOffice.org,[126] which allow users to choose one of two or more licenses to comply with. One of the licenses is usually GNU General Public License or some other model Open Source License and it can be easily accepted by Open Source developers. For those who are afraid of the "copyleft virus" some other license is presented and they can thus merge the product with their intellectual property without taking the risk of having to disclose it publicly. The additional strategy of project originator may be to issue the same code as proprietary, perhaps with some enhancements which are lacking in the Open Source version. This is not fully in-line with Open Source ideals, but the economic goal is clear: multiple licensing allows the use of the code by the biggest number of developers and is one of the ways in which commercial companies secure for themselves the advantages of "network effects".

Companies also use Open Source as their competition strategy, to prevent others

[125] For both new Mozilla source files and modifications to existing source files, triple-license scheme is required, which allows use of the file under the terms of any one of the MPL, GNU GPL, version 2 or later, or GNU LGPL, version 2.1 or later. For modifications to existing files already under the license in question use of the file under the terms of either the MPL or the GNU GPL must be allowed. The choice of the license is given to the licensee. Still, some of the Mozilla source files are licensed under the terms of other licenses, such as the BSD. (The Mozilla Organization, *mozilla.org License Policy*, available at: http://www.mozilla.org/MPL/license-policy.html)

[126] OpenOffice.org uses a "dual-licensing scheme" for source-code contributions: the GNU LGPL and SISSL. A separate documentation license, Public Documentation License (PDL), is used for most documents published on the website without the intention of being included in the product. When the GNU GPL license is used, the libraries and component functionality of the OpenOffice.org source code are licensed under the GNU LGPL. (Sun Microsystems, *License Page*, available at: http://www.openoffice.org/license.html)

from gaining monopolistic position in the product market. Apart from giving their products away free of charge, they open its code, thus gaining even more publicity. Some try to take it even further and perceive Open Source as a way to secure market position for themselves. Indeed, one clear result of "copyleft" clauses is that everything a competitor would add to the software has to return to the originator of the project anyway. This allows keeping the platform for competition and gaining the edge over other players by providing additional services for the software. A similar idea – to establish an industry standard – is being pursued by Sun with their SISSL, which is a license requiring the publication of all modifications to the licensed standard. Thus, the company which has set the standard can observe how it develops and makes profit, for example, by selling proprietary software basing on the standard.

3.4 Model Open Source Licenses

Out of various factors influencing the role of law in the Open Source Movement, one more specific approach towards law has to be discussed. Apart from the standardization of the licenses owing to the introduction of FSD and OSD, the evolution of model licenses has been a noticeable trend. The originators of new Open Source projects usually do not draft a completely new license but rather decide to publish their code under one of the already known and respected contracts, which are published on the Internet by their drafters and maintained in a similar way to the model laws of such organizations as WIPO or UNCITRAL. The model licenses are so widely known that they are usually referred to by an abbreviated name, such as "GNU GPL" already mentioned few times in this paper.

What follows is a brief presentation of some of the model licenses, which at the same time correspond to the Open Source projects that have been selected for the discussion in this paper. Apart from underlying the phenomenon of the existence of model licenses, the purpose of this presentation is to show how different they are, while still considered in compliance with FSD and OSD.

The particular clauses of the licenses will be invoked in the next Chapter while presenting and evaluating their compliance with various laws and overall legal impact. For a very detailed comparison the reader may be referred to the publication of French Agency for the development of electronic administration (ADAE),[127] which points out as much as ten rights usually covered by Open Source licenses and compares how the most popular ones deal with these rights.

3.4.1 GNU General Public License

GNU General Public License (GNU GPL) is the most widely known model license, commonly associated with Open Source. It was drafted by FSF and has been developed by them to reach its current version 2 in 1991. It is considered to fully comply with FSD and employs the copyleft concept.[128] It has also been accepted by OSI as complying with OSD.[129] According to the statistical data available at SourceForge.net, roughly 70% of Open Source Software is licensed under GNU GPL, whereas only the popularity of GNU Lesser General Public License exceeds 10%.[130] Although both of these licenses were drafted under the influence by Stallman and are copyrighted by FSF, not only the software belonging to the GNU Project is licensed under their terms.

[127] Agency for the development of electronic administration (ADAE), *Guide to Choosing and Using Free Software Licenses for Government and Public Sector Entities*, available at: http://www.adae.gouv.fr/upload/documents/free_software_guide.pdf.

[128] Free Software Foundation, *GNU General Public License*, available at: http://www.fsf.org/licenses/gpl.txt.

[129] GNU GPL is even expressly addressed in the annotation to OSD Sec. 9 "License Must Not Restrict Other Software" and declared to be compatible with this requirement.

[130] Open Source Technology Group, *SourceForge.net: Software Map*, available at: http://sourceforge.net/softwaremap/trove_list.php?form_cat=14.

GNU GPL consists of preamble, precise terms and conditions and "How to Apply These Terms to Your New Programs" section. The terms and conditions part contains 13 sections numbered from 0 to 12, which cover: definitions; rights to copy and distribute verbatim copies of source code; rights to modify and distribute modifications of source code; rights to copy and distribute object code; "copyleft"; assent by conduct; original licensor grant; conflicts with other legal obligations; the right of FSF to update GNU GPL; special conditions of combining programs licensed under other licenses; warranty disclaimer and liability limitation.

To put it in a nutshell, GNU GPL allows the licensee to use, modify and distribute the program in source or object code for any purpose. However, it requires that the access to source code be always possible. What is more important, by the "copyleft" clause in Sec. 4, GNU GPL forbids licensee to "copy, modify, sublicense, or distribute the Program except as expressly provided under [GNU GPL]". This clause attempts to make any program once released under GNU GPL remain distributed under its terms whoever distributes it. "Copyleft" by virtue of Sec. 2 b) extends to "any work ..., that in whole or in part contains or is derived from the Program or any part thereof".

For those who would like to make concessions to the proprietary software world, FSF introduced GNU Lesser General Public License (GNU LGPL), which main feature is that it allows linking with other proprietary program.[131] It is designed to cover software libraries, which are collections "of software functions and/or data prepared so as to be conveniently linked with application programs to form executables."[132] It is described by FSF

[131] Free Software Foundation, *GNU Lesser General Public License*, available at: http://www.fsf.org/copyleft/lesser.txt.
[132] *Id.*

as "not a strong copyleft license, because it permits linking with non-free modules."[133]

Interestingly, GNU GPL itself is copyrighted by FSF, which allows for copying and distributing verbatim copies of it but not for changing. It is an obvious attempt to promote standardization in Open Source licensing by making GNU GPL the model contract. Indeed, the majority of developers simply put notices of GPL in their programs, according to the "How to Apply These Terms to Your New Programs" section. For example, Linux kernel is released under the terms of GPL, preceded by the following statement of Torvalds:

> NOTE! This copyright does not cover user programs that use kernel services by normal system calls - this is merely considered normal use of the kernel, and does not fall under the heading of "derived work". Also note that the GPL below is copyrighted by the Free Software Foundation, but the instance of code that it refers to (the Linux kernel) is copyrighted by me and others who actually wrote it.[134]

3.4.2 BSD License

BSD License[135] has initially applied to BSD operating system distributions issued by the University of Berkeley. Later on, after a small modification[136] it has become a model license agreement and was adopted by the developers of FreeBSD, NetBSD, OpenBSD and other projects not necessarily connected with the University. According to SourceForge.net, it is now the third most popular Open Source license after GNU GPL and GNU LGPL.[137]

[133] Free Software Foundation, FN 99.

[134] Linus Torvalds, *Note to Linux Kernel*, available at: http://www.linux.de/linux/gnu.html. The note also states that the only valid version of the GPL as far as the kernel is concerned is this license (*i.e.* v2), unless expressly otherwise stated.

[135] The text of BSD License may be found at: http://www.xfree86.org/3.3.6/COPYRIGHT2.html.

[136] The original BSD license included the following introductory clause, which is not present in the model license:
Copyright (c) 1993 The Regents of the University of California. All rights reserved.
This software was developed by the Computer Systems Engineering group at Lawrence Berkeley Laboratory under DARPA contract BG 91-66 and contributed to Berkeley.
All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by the University of California, Lawrence Berkeley Laboratory.

[137] Open Source Technology Group, FN 130.

In comparison with GNU licenses, BSD is a very short contract. FSF describes it as "a simple, permissive non-copyleft free software license, compatible with the GNU GPL."[138] The BSD License permits redistribution and use in source and binary forms, with or without modifications. The licensee is obliged to retain the original author's copyright notice, conditions of the license itself, the disclaimer of warranties and damage limitation. Additionally, the use of the author's name to endorse or promote products derived from the software covered by the license is prohibited without specific prior written permission.

3.4.3 Apache Software License

Until February 2004, the Server was available for downloading by everyone under the terms of Apache Software License, Version 1.1.[139] It was a short contract, allowing redistribution and use in source and binary forms, with or without modifications. Licensee deciding to redistribute, however, had to retain the copyright notice of the Apache Software Foundation, the list of conditions of the License and the notice stating: "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)."[140] The License required prior permission for the use of "Apache" in the names of the products derived from the server. It also included disclaimer of warranty and liability. Thus, it was almost the exact copy of BSD License.

In February 2004, the Apache Group introduced a new license – Apache Software License, Version 2.0. It is substantially longer now, contains nine sections and an appendix explaining how to apply the license. The sections cover: definitions; grant of copyright and patent license; redistribution right; coverage of subsequent contributions; trademarks;

[138] Free Software Foundation, FN 99.
[139] The Apache Software Foundation, *Apache License Version 2.0*, available at: http://www.apache.org/licenses/LICENSE-2.0
[140] *Id.* at Sec. 3.

warranty disclaimer and liability limitation; possibility of accepting warranty or liability. The patent license grant in Sec. 3 contains a termination clause in case of the licensee's patent litigation alleging that the licensed program constitutes patent infringement. This patent license termination clause is considered by FSF the reason for the incompatibility between GNU GPL and the Apache License.

The Apache License allows to reproduce licensed program in source or object code and it is also possible to publish the modifications or derivative works under a different license. One should note that the Apache License contains its own definition of "derivative work". The license grant contains permission to sublicense the program, but there is no explicit mention that the redistribution of the originally licensed program should be covered by the Apache License or any other particular terms. Thus, it does not contain an express "copyleft" clause.[141] However, the redistribution conditions demand, *inter alia*, to give the recipients the copy of the Apache License and to retain copyright and other notices (Sec. 4 a. and 4 c.). It is allowed to add one's own attribution notices within derivative works, provided that such additional attribution notices cannot be construed as modifying the License (Sec. 4 d.). Since the distribution under different license is allowed for modifications and derivative works only, it can be argued *a contrario*, that it is prohibited with regard to the original program. Thus, it may be concluded that the Apache License, Version 2.0 contains a weak "copyleft" clause.

3.4.4 Mozilla Public License

Mozilla Public License, Version 1.1 is one of the licenses under which the software developed in Mozilla Project is released. The license is classified by FSF as "not a strong

[141] FSF classifications remain silent as to whether the license is a "copyleft" or not.

copyleft" free software license incompatible with the GNU GPL.[142] MPL is quite complex and addresses many issues that the previously discussed licenses did not expressly regulate. It contains thirteen sections, which cover: definitions; rights to the source code; distribution obligations; conflict with other legal obligations; the license subject matter; the right of Mozilla Foundation to issue new versions of the license; warranty disclaimer; termination; liability limitation; the rights of U.S. government end-users; miscellaneous provisions; responsibility distribution among the parties; the possibility of multiple-licensing. Exhibit A to the license contains a notice to be placed in files distributed under it.

Similarly to Apache License, MPL grants both copyright and patent license. It distinguishes between "initial developer" and "contributors" and covers their license grants in separate subsections (Sec. 2.1 and Sec. 2.2 respectively). The license addresses third party claims and requires contributors to notify about these they know of in a special file (Sec. 3.4. (a)). MPL allows licensees to distribute program not in the source code and allows such distribution to be subject to different licenses, which is intended to make possible the integration of MPL-covered code into other projects and even proprietary software. Such different license may, for example, include different payment or support terms and even use restrictions.[143] Miscellaneous provisions in Sec. 11 contain, among others, choice of Californian law clause and choice of Californian courts. They also expressly exclude the application of UN Convention on Contracts for the International Sale of Goods but are silent on the applicability of UCC Art. 2. Additionally, they exclude the applicability of "[a]ny law or regulation which provides that the language of a contract shall be construed against the

[142] Free Software Foundation, FN 99.

[143] The Mozilla Organization, *Annotated Mozilla Public License*, Annotation to Sec. 3.6, available at: http://www.mozilla.org/MPL/MPL-1.1-annotated.html.

drafter."

MPL contains "copyleft" clause (Sec. 3.1) different from the one included in GNU GPL. The scope of it is delimited by the understanding of "Modifications" defined in Sec. 1.9.[144] and further explained in MPL-FAQ, which states that modifications are "[a]ny changes to MPLed files, or new files into which MPLed code has been copied ... . New files containing only your code are not Modifications, and not covered by the MPL." It is important to note that MPL does not refer to "derivative works", a copyright law term of art, in setting the scope of its "copyleft".

3.4.5 Sun Industry Standards Source License

Sun Industry Standards Source License, Version 1.1 (SISSL)[145] has been drafted by Sun Microsystems and is used, for example, in the development of their OpenOffice.org Open Source project. It is the least popular model license out of discussed in this paper but nonetheless triggers interesting legal issues. Sun has issued some other standard license agreements, such as Sun Public License, Sun Community Source License or Sun Solaris Source Code (Foundation Release), Version 1.1 License.[146]

SISSL is very much similar to MPL and some sections are even copied word-by-word but there are differences. The most important one concerns the "copyleft" clause included in SISSL, which is different from these in GNU GPL or MPL. It is spelled out in

[144] MPL Sec. 1.9 "Modifications" means any addition to or deletion from the substance or structure of either the Original Code or any previous Modifications. When Covered Code is released as a series of files, a Modification is:
A. Any addition to or deletion from the contents of a file containing Original Code or previous Modifications.
B. Any new file that contains any part of the Original Code or previous Modifications.

[145] Sun Microsystems, *Sun Industry Standards Source License – Version 1.1*, available at: http://www.openoffice.org/licenses/sissl_license.html.

[146] The last two licenses are declared by FSF as non-free software licenses.

Sec. 3.1 and requires among other things, the modifications[147] to comply with Standards set out in Exhibit B to the license. If the modifications do not meet such requirements, SISSL imposes on the licensee the obligation to publish information necessary to identify the differences with Standards. As it is explained in FAQ, "[SISSL] allows the user to do what they like with the source base, modify it, extend it, *etc.*, but [compatibility must be maintained]."[148] At the same time, SISSL as well as MPL allows for integrating the licensed program into a larger work and issuing such work under different license.

3.5 Intellectual Property Rights Management in Open Source Movement

In the case of proprietary software, the assignment of copyright to one entity is common. Rights in employees' contributions are often transferred to their employers in employment contracts or even by the law itself. Assignments between cooperating developers are extensively covered in their contracts. Open Source is a completely different software development model and precisely because of that its legal situation could often be much more complicated. As it was already indicated it is customary for every project to have at least one coordinator, who is considered by the community to "own" the project, *i.e.* to have power of deciding about its development; what to change or add in the program. Above all coordinators have unquestioned power of releasing newly developed versions of their projects.

Legally speaking; however, in the absence of any agreement to the contrary, the assignment of copyrights is governed by the applicable copyright statutes. Thus, if a contribution fulfills copyrightability requirements the rights in it should be attributed to the particular contributor. If it forms a part of a bigger work it may also be the case that this work

[147] The definition of "Modifications" in SISSL is almost exactly the same as the one in MPL.
[148] Sun Microsystems, *FAQs*, available at: http://www.openoffice.org/FAQs/faq-licensing.html.

is subject to joint copyright of all contributors, whereas the producer could be considered to hold copyrights to the whole. Logically then, the producer's copyrights should be attributed to the project coordinator but even in such a case, there is the issue of separate copyrights in particular contributions. For example, the most important piece of Open Source Software – the Linux kernel is "copyrighted by [Linus Torvalds] and others who actually wrote it",[149] which means the necessity of considering copyrights of many hackers from around the world, although only few persons take the major decisions concerning development.

Instead of relying on the coordinator's authority and moral norms only, contributors to an Open Source project may transfer their rights to one trustworthy entity or retain but expressly license them. For example, Free Software Foundation encourages developers to transfer copyrights in their respective projects to the foundation, which certainly helps to clarify legal situation of Open Source projects in general. For some time now, many Open Source projects' coordinators have also started employing some intellectual property rights management schemes. This is good, because the lack of diligence may result in infringement of contributors' or third parties' rights, which may happen if there is no legally valid contribution (assignment of rights or license) or its scope does not encompass the publication of the source code. In the worst case the contributed code may in fact belong to someone else, such as the contributor's employer by the virtue of the employment contract.

Thus, in the case of major Open Source projects, instead of the coordinator being a natural person, there is some kind of entity formed, such as the Apache Software Foundation or Mozilla Foundation.[150] The Apache Project has evolved from the loose cooperation of

[149] Torvalds, FN 134.

[150] Similar entities stand behind such purely non-profit projects as Debian (Software in the Public Interest, Inc.) or Slackware (Slackware Linux, Inc.)

contributors exchanging “patches” to the original NCSA server but currently it employs quite sophisticated system for managing intellectual property rights of project participants. This system is based on the hierarchical project participants organization. Although everyone's contributions are welcome, only “commiters” and those higher in the structure are allowed the write access to the code repository. To be a “commiter” one has to sign a Contributor License Agreement (CLA).[151] The CLA covers all past, present and future contributions to the project and grants to the Apache Foundation the license to use, execute, prepare derivative works and distribute them.

The farther reaching approach is visible in the case of Open Source projects coordinated by commercial companies. For example, contributors to the OpenOffice.org project have to submit the Joint Copyright Assignment form (JCA).[152] JCA is not a license but a copyright assignment and purports to “assign to Sun Microsystems joint ownership in all worldwide common law and statutory rights associated with the copyrights, copyright application, copyright registration and moral rights in the contribution to the extent allowable under applicable local laws and copyright conventions.”[153] However, assignor retains the right to use the contribution for his own purposes.[154]

Whichever approach is taken in a particular project, it is important that the coordinator at all times knew which part of the code is copyrighted by whom and what is the

[151] The Apache Software Foundation, *Individual Contributor License Agreement*, available at: http://www.apache.org/licenses/cla.txt

[152] Sun Microsystems, *OpenOffice.org Open Source Project Joint Copyright Assignment by Contributor*, available at: http://www.openoffice.org/licenses/jca.pdf.

[153] JCA Sec. 2. Before introducing JCA, Sun required from contributors to transfer all copyrights, just as it is the case with FSF and contributors to the GNU Project. According to the Sun’s statement published on the OpenOffice.org FAQ Web page (FN 148), signing JCA by the contributor who previously signed copyright assignment form supersedes this agreement. Sun also holds that there is no process to cancel JCA in order to protect the code base so everyone who uses it can depend on its continued functionality; however, ownership of the contributions that were not accepted and included in the suite “reverts” to the contributor.

[154] JCA Sec. 2

legal basis for the code's inclusion. The least what could be done is to insist on the contributions being expressly licensed to the project under one of the model Open Source licenses. The supply of legal code obviously lies in the coordinator's interest because the users, endangered by infringement suits instigated by the lawful copyright holders, would quickly turn away from the project. But it is more important that the negligence of the coordinator could result in the loss for the whole Open Source community, even if he himself was protected by warranty disclaimers. Fewer hackers would be willing to contribute and there would be no clear guidance whether it is possible to fork the project without risking the inheritance of illegal code. On the other hand, this may not be so burdensome in every case, because the public availability of source codes is a big help in determining whether there really was an infringement. Still, there is no reason why the Open Source project coordinator should scrutinize the inflowing code less meticulously than proprietary software developer.

3.6 Evaluation of Role of Law in Development of Open Source Software

To sum up, the most important legal difference between proprietary software and Open Source is the scope of rights given to the users. It is basically true that developers of the former limit users' rights in order to secure market position for themselves, whereas the latter are community-oriented and the system designed by them serves all the players. But the analysis contained in the previous sections shows that this is a far-reaching generalization. In fact, although Open Source Movement as a whole may be considered the opposite of proprietary software, it is not homogeneous. Its supporters differ as to their ideologies, motivations, organization structures and, as an implication, the use they make of law. There are moralists, such as Stallman and pragmatists represented mainly by OSI. Other division lines run between altruists merrily participating in the hacker's culture and profit-driven commercial companies designing various Open Source marketing models.

Somehow, all of them are able to work together and produce the software that still can be called “Open Source”. To certain extent it is possible because FSF and OSI are quite flexible in the scrutiny of licenses, although the former is backed up with a lot of ideology whereas the latter is presented as completely pragmatical. Both of them; however, agree on a common set of core rules for all the players involved in the Open Source Movement, which oblige to respect users' rights. They can be ideologized or commercialized, but not limited. Furthermore, the evolution of model licenses and the practical dominance of GNU GPL also serve as the movement's integrators. Apart from particular developers' savings on the legal fees for writing a new license, the use of model ones contributes towards legal certainty, standardization and allows for the cooperation and distribution of the software to be executed faster.

The hackers follow specific social norms and treat legal action as the last resort. This is one more generalization, because the cooperation between hackers is often highly formalized and the coordinators of major Open Source projects have provided for quite sophisticated intellectual property rights management schemes, whereas even the smaller projects make much ado about their use of, say, GNU GPL. The main aim is to protect the community from the threat of proprietarization of the code and to allow the continuation of the moral crusade for the free software for all. On the other hand, commercial companies have found ways of making the legal use of Open Source Software to gain market position and even to maintain concessions for the still popular proprietary model. For them, licensing is of much importance too.

In the end, the preceding discussion supports the most important conclusion on the role of law in the development of Open Source Software, which is that the law as used by Open Source Movement protects interests of individual users and does not leave them at the mercy of the sellers by making just mere consumers out of them, as it is the case with the law and technology of proprietary software.

Chapter 4 - Legal Controversies of Open Source Licensing

From the legal point of view, the major goals of the participants in Open Source Movement presented in the previous Chapter are realized with the use of specific software licenses, such as the model GNU GPL. Logically, the next issue that will be discussed is whether these licenses are valid, enforceable and whether they can be executed according to the intent of the licensors. In other words, the current Chapter is devoted to the analysis of the major legal controversies surrounding Open Source Movement, which have been identified in the course of research. Apart from analyzing threats, it also contains discussion on the consequences of the licenses' enforceability and attempts to contribute towards recently heated debate on software patents.

4.1 Contract Law and Open Source Licensing

As it has already been described, the issue of legal effect of Open Source licensing is a part of the wider debate concerning "-wrap" contracts in the electronic age, which have developed among legal scholars and practitioners for some time now. To the best knowledge of the author of this paper, in no legal system are there any special provisions governing the validity and enforceability of Open Source licenses and few courts have looked at this particular issue so far. The only court decision concerning an Open Source license was rendered in Germany.[155] In this case, the court confirmed preliminary injunction issued against a company that had infringed GNU GPL by not publishing source codes of their distributed modifications to an Open Source program. The decision contains numerous

[155] Landgericht München, 19.5.2004, 21 O 6123/04, unofficial English translation available at: http://www.groklaw.net/article.php?story=20040725150736471. The plaintiff in this case has reached settlements in similar disputes before, without having to resort to the court. See. *e.g.* Harald Welte, *netfilter project GPL settlement with Securepoint*, available at: http://www.netfilter.org/news/2004-03-25-securepoint-gpl.html.

remarkable holdings of the court, starting from the conclusion that the issuance of software under GNU GPL does not constitute a waiver of copyright. Moreover, the court found that a web page reference to the publicly available terms of license constitutes a valid mean of incorporating these terms to the parties' contract. This is particularly important when compared to the holding in the American case *Specht v. Netscape*, presented below.

The German decision contains the material analysis of GNU GPL clauses as well. The court concluded that the automatic termination of rights upon the licensee's infringement provided for in GNU GPL Sec. 4 is a permissible limitation of rights under German Copyright Law because neither it affects third parties' rights (sublicensees), nor the marketability of the licensed rights. It is also not too burdensome for the infringing party itself, as the rights under the license can be reacquired at any time by the acceptance and compliance with GNU GPL. Moreover, according to the court if there were bases for finding invalidity of a clause or the whole GNU GPL then it should be concluded that no agreement has been reached. As the consequence, any use of the software would be illegal and no danger of proprietarization or closing the code would arise. Saying so, the court expressly supports the rationale of GNU GPL Sec. 5 and of Moglen, whose position is that there are no gains from infringement, which would not exist if the license were followed.[156]

In the U.S., where Open Source licenses originated, so far they have been mostly subject to academic discussion. The *USL v. BSDI* case presented above ended with settlement before the court had the possibility to consider its merits. Recent litigation, *Progress Software Corp. v. MySQL AB*[157] directly concerned breach of GNU GPL license but was settled too. Before settlement, the court agreed that much depends on the meaning of

[156] Moglen, FN 120 and FN 121.
[157] 195 F. Supp. 2d 328 (D. Mass. 2002) (preliminary injunction).

"derivative works" under GNU GPL, but when issuing preliminary injunction it did not rule on whether the Gemini software linked to MySQL was covered by the license. The merits are likely to be reached in the *SCO v. IBM*[158] dispute, which is to be decided in 2005. Briefly speaking, the parties in this case are two companies that worked together on the development of a proprietary UNIX-like system. The defendant has allegedly contributed their proprietary improvements to the Linux kernel. At the time of this writing, it was too early to determine whether the validity of GNU GPL would be one of the issues of the case.

There is one U.S. court decision, *Planetary Motion, Inc. v. Techsplosion, Inc.*,[159] concerning trademark infringement of CoolMail program, which contains some *dicta* on GNU General Public License. The court treated this license favorably, saying that "[s]oftware distributed pursuant to such a license is not necessarily ceded to the public domain and the licensor purports to retain ownership rights."[160] According to the court, "the license itself is evidence of [the copyright owner's] efforts to control the use of the "CoolMail" mark in connection with the Software."[161]

Thus, in the absence of any tailored statutes or elaborate case law, the existing law has to be relied upon. At the very least, the applicability of the legal rules discussed in the following subsections should be considered.

4.1.1 U.S. Case Law on Software Licensing

American courts deal with the issue at least from the early 90s. One of the first decisions, *Step-Saver Data Sys., Inc. v. Wyse Techn.,*[162] concerned a "shrink-wrap" software

[158] *The SCO Group, Inc. v. International Business Machines*, No. 2:03cv0294 (Plaintiff's Amended Complaint) (D. Utah, filed June 16, 2003).
[159] 261 F.3d 1188 (11th Circ. (Fla.), 2001).
[160] *Id.* at 1198.
[161] *Id.* at FN 16.
[162] 939 F.2d 91 (3d Cir. 1991).

license and did not look at it favorably. The court refused to treat "shrink-wrap" license terms as incorporated into the parties' contract. The terms were not expressly agreed upon, only referred to by the notice on the box, stating that opening constitutes assent. The court found that the contract was formed over the phone and treated license as additional terms materially altering the agreement, which pursuant to UCC 2-207(2) do not become part of the contract. The court refused to consider the vendor's acceptance conditional upon the buyer's assent to the "shrink-wrap" license, because the vendor did not show enough "unwillingness to proceed with the transaction unless the additional or different terms are included in the contract."[163] It must be noted, however, that before shipment the vendor assured that the "shrink-wrap" license would not apply in this particular case, and its terms were substantially different from those negotiated by the parties before the transaction.

"Shrink-wrap" licenses accompanying the sale of computer programs in retail stores were more favorably looked upon in the landmark case *ProCD, Inc. v. Zeidenberg*.[164] The issue presented to Judge Easterbrook was whether the defendant breached the contract by making commercial use of software, which was expressly prohibited by the license. The court noted that the installation and use of the program was possible only after the acceptance of the license, that there was a reference to it outside of the box and that there was refund possibility if its terms were found unacceptable.[165] Thus, the court held that one of the terms to which the buyer agreed by purchasing the software is that the transaction was subject to a license.[166] Judge Easterbrook strengthened his analysis by comparing "shrink-wrap" contracting with other transactions where payment precedes making all terms known to the

[163] *Id.* at 103.
[164] 86 F.3d 1447 (7th Cir. 1996).
[165] *Id.* at 1452.
[166] *Id.* at 1450.

buyer. Moreover, he treated such method of concluding contracts as less troublesome than the negotiation of all terms beforehand and allowed them for economical reasons.

Further guidelines on how the contracting procedure should look like for the license to bind the end-user may be found in *M.A. Mortenson Co. v. Timberline Software Corp.*[167] The court repeated after *ProCD*, that money-now-terms-later agreements are common and it is also generally expected for software to be covered by license.[168] The court relied on UCC 2-204 which allows parties to form a contract in "any manner sufficient to show agreement" and found such agreement with license terms manifested by the installation and use of software by the buyer. As a result, every license term which was legal and "not found to be unconscionable" was held to bind the parties, even if it were not individually negotiated beforehand.[169] The court attributed much importance to the fact that the notice of license was printed outside the box, on the hardware supplied together with software and presented on the screen during the use of the program.

"Click-wrap" agreements were put by the U.S. courts under a similar scrutiny. In *Hotmail Corp. v. Van$ Money Pie, Inc.*[170] making available of e-mail account for users after their clicking on the "I agree" button under the terms of use presented on screen was found sufficient for the court to issue a preliminary injunction against actions expressly prohibited in these terms. Similarly, in *Caspi v. Microsoft Network, L.L.C.*[171] the court enforced the "click-wrap" contract because adequate notice and enough time to review and withdraw was provided.

Some attention was given to the form of communicating assent in *Specht v.*

[167] 970 P.2d 803 (Wash. Ct. App. 1999), aff'd, 998 P.2d 305 (Wash. 2000)
[168] 970 P.2d 803, 808
[169] *Id.* at 809.
[170] 47 U.S.P.Q.2d 1020 (N.D. Cal 1998).
[171] 732 A.2d 528 (N.J. Super Ct. App. Div. 1999), cert. denied, 743 A.2d. 851 (1999).

Netscape Communications Corp.[172] and in *Ticketmaster Corp. v. Tickets.com, Inc.*[173] Both concerned "browse-wrap" agreements. In *Specht* the download page for the program at issue contained notice merely asking users to review and agree to the terms of software license before downloading and installing. The actual license text was accessible only after the user scrolled down the page and clicked on the link provided there, but it was possible to download, install and use the program without doing this. The request to read and accept the license was not treated by the court as a condition but as an invitation only. As there was no assent communicated in any way, the contract was held not to include additional terms contained in the license. Similarly, in *Ticketmaster*, there was no binding contract found by the court in the case when terms of use were hyperlinked to in the small print and users were not forced to accept it before accessing the web page.

The U.S. courts; however, seem not to have decided where to draw the borderline. For example, a "browse-wrap" agreement was enough for the court to issue a preliminary injunction in *Register.com, Inc. v. Verio, Inc.*[174] Even if there was no clear communication of assent, such as clicking on an "I agree" button, the presentation of the terms of use together with the results of the service (WhoIs database query) made the court conclude that it was "likely" for a "browse-wrap" contract to have been concluded. In an earlier case, *EF Cultural Travel BV v. Zefer Corp*[175] the court pointed out that it is possible for a website owner to bind users with his terms of use by saying so on the webpage or providing for a clearly marked link to them. A software license concluded in a similar way, which granted rights to create and distribute additional levels of a popular computer game, was held to bind parties in

[172] 306 F.3d 17 (2nd Cir. (N.Y.) 2002).
[173] 54 U.S.P.Q.2d (BNA) 1344 (C.D. Cal. 2000).
[174] 2004 WL 103400, 69 U.S.P.Q.2d 1545 (C.A.2 (N.Y.), 2004).
[175] 2003 U.S. App. LEXIS 1336 (1st Cir. 2003).

Microstar v. Formgen, Inc.,[176] although neither reading nor accepting it was the precondition of using the level editor program. The license was contained in LICENSE.DOC file on the disk together with other program's files and was only referred to in the opening screen of the program. It was not presented on the computer screen, there was no notice on the box and no fixed period to return and withdraw. The defendant argued that no average user would access the license file, but the court did not find this persuasive. At the same time the court noted that movants could have done "a better job" in ensuring the users' assent to the license.[177] These three cases do not fit neatly in the previously described case law, but the courts there also paid much attention whether the intent to bind with certain terms was clearly communicated (as in *EF Cultural Travel*) or whether the user knew about their existence. For example, it seems to have been decisive in *Register* that the defendant used the service for a number of times and in *Microstar* that the knowledge of the license was actually admitted by the licensee.

The existing case law does not provide clearcut rules and it has to be kept in mind that the issue of enforceability and validity of "-wrap" contracts is still unresolved.[178] However, it seems agreed that in order to secure high probability of contract enforcement, the licensor should provide for clear communications between the parties. Namely, the existence of the license must be made known to the buyer at the time of purchase (*e.g.* by a notice on the box). Also his assent must be clearly communicated (*e.g.* clicking on an "I agree" button, mere availability of license on a linked page, as in *Specht*, is not enough). If these conditions

[176] 942 F. Supp. 1312 (S.D. Cal. 1996), aff'd in part, rev'd in part on other grounds, 154 F.3d 1107 (9th Cir. 1998).

[177] 942 F. Supp. 1312, 1318.

[178] For more caselaw analysis see: Kevin W. Grierson, *Enforceability of "Clickwrap" or "Shrinkwrap" Agreements Common in Computer Software, Hardware, and Internet Transactions*, 106 American Law Reports 5th 309.

are fulfilled, the U.S. courts are likely to hold "shrink-wrap" and "click-wrap" agreements valid and enforceable, although their terms are not available to the seller at the time of payment or even delivery. Still, there are additional important factors that may be taken into account by the courts, such as failure to return the purchased software for refund after the term indicated in the license, to mention only one of them.

4.1.2 UCITA and Relevant European Union Directives

The inadequacies of existing case law rules for the reality of contracting in the electronic age together with many doubts resulting from the application of UCC Art. 2 (Sales of goods) to software licenses (discussed in subsequent section) has triggered attempts to develop a special statute. At first there was the proposal of a new UCC Article – 2B (licenses), but in the late 90s it transformed into a separate Uniform Computer Information Transactions Act (UCITA). It is presented now as a coherent and uniform body of law for computer information transactions, including software licenses. UCITA provides for thorough definitions and delimitation of its scope (Part 1), regulates contract formation and terms (Part 2), its construction (Part 3), warranties (Part 4), transfer or interests and rights (Part 5), performance (Part 6), breach of contract (Part 7) and remedies (Part 8). The last Part (Part 9) contains miscellaneous provisions.

For the contract formation, UCITA adopts so called "layered" or "rolling" contracts theory, which does not treat contracting as a single event.[179] It relies on cases such as *ProCD* or *MA Mortenson* that recognize that contracts are often formed over time by the parties, whose expectations are that terms will follow or be developed after performance begins.[180]

[179] Robert Hillman, *Rolling Contracts*, 71 FORDHAM LAW REVIEW 743 (2002).
[180] UCITA Sec. 208 cmt. 3 (Approved Official Draft), available at: http://www.law.upenn.edu/bll/ulc/ucita/2002final.htm.

Pursuant to UCITA Sec. 209(a) a mass-market license is only effective if the licensee agrees to it, such as by manifesting assent, before or during initial use or access to program. An elaborate understanding of "manifesting assent" is given in Sec. 112. However, a term may not be binding even if agreed upon when, for example, there was no opportunity to review it before agreeing or it was unavailable after the agreement (in nonelectronic or electronic format capable of being printed or stored for review).[181] Thus, hiding the license in small print or granting the access to program in an malicious attempt to induce infringement would not be protected practices.

UCITA would make most of the software licenses enforceable but only Virginia and Maryland adopted it and many other states decided to prevent choice of law clauses resulting in the application of UCITA by making them null and void.[182] For those jurisdictions where UCITA is not the law, it may be helpful that the court in *Specht* concluded that under either UCC Art. 2, common law or UCITA the result would be the same.

The validity and enforceability of software licenses in Europe have to be evaluated for each state separately and in the light of its national law. American law may play some role to the extent the licenses contain choice of law clauses, such as these included in MPL or SISSL. There are no harmonized provisions at the level of the European Union, which would refer directly to software licenses, but there are many consumer protection acts that have to be taken under consideration. Some of them are discussed in the section on the liability of Open Source developers.

European directives, which may be referred to at this point are: the Directive on the

[181] UCITA Sec. 209(a)(1)-(4).
[182] Grierson, FN 178.

protection of consumers in respect of distance contracts[183] and the Directive on certain legal aspects of information society services, in particular electronic commerce, in the Internal Market (E-Commerce Directive).[184] Unfortunately, they are not tailored for mass-market licenses and seem better fit for more traditionally concluded contracts by the exchange of offer and acceptance. Moreover, they are not intended to regulate the formation issues discussed in this section. However, they include some rules relevant for the construction of contracts, such as the requirement to provide the consumers with adequate prior information. Additional particularly interesting obligation is the one of the suppliers to make their commercial communications clearly identifiable as such by the suppliers (E-Commerce Directive Art. 6 (a)).

4.1.3 Applicability of Sales of Goods Laws

Computer programs, as all other information, are intangibles. They certainly may be embodied in a fixed medium, such as a CD or hard disk, but still the protection of intellectual property laws has as its subject the effect of the author's creativity, not the movable fixation. On the other hand, the program, as any other copyrighted work, after its fixation becomes a tangible thing and may be subject to the sales contract. Software licensing agreements; however, usually expressly state that the software is not sold but licensed only. The title to the program is not transferred to the buyer but he is merely granted with some limited rights, such as the right of use. There seems to be much difficulty in determining what exactly forms the subject of software transactions and which law is to be applied, especially in jurisdictions having special provisions regulating sales of goods.

In the international context, the applicability of UN Convention on Contracts for the

[183] 97/7/EC, OJ L 144, 04/06/1997, P. 19.
[184] 2000/31/EC, OJ L 178, 17/07/2000, P. 1.

International Sale of Goods (CISG) must be taken under consideration. At first glance CISG may be considered applicable, as many software transactions are concluded between persons having places of business in different states (CISG Art. 1) and especially in the trade in Open Source Software many programs are acquired not for personal, family or household use (expressly excluded by CISG Art. 2(a)). The final determination on CISG applicability depends mainly on the understanding given to "contracts of sale of goods", which constitute the subject matter of CISG (Art. 1) but are nowhere explicitly defined. The discussion on whether the license constitutes sale resembles the one on UCC Art. 2 presented below. Oosterbaan raised that even if no intellectual property rights are sold and despite software is intangible in any case, the fact that in practice licenses are granted for unspecified period of time and for a single payment together with the sale of a copy allows to conclude that the parties enter into a sales contract.[185] Moreover, it is agreed that if CISG applies to transactions in software embodied in tangible media then there is no reason why electronically transferred software should be excluded.[186] Diedrich concludes that "any item that can be commercially sold and in which property can be passed on and which is not explicitly excluded from the CISG's sphere of application"[187] constitutes a good. Not all Open Source developers are aware of that and only few of them draft their licenses to expressly exclude the applicability of CISG – the examples are MPL and SISSL.

The effect of CISG's applicability is crucial for contract formation and parties' obligations determinations. For example, whether the "shrink-wrap" or "click-wrap" license

[185] Dinant T. Oosterbaan, *Shrink-wrap License Agreements in the Netherlands*, in: ERIK MOSESSON ED., SOFTWARE PROCUREMENT NORDIC YEARBOOK OF LAW AND INFORMATICS 1992, 105, 107-108 (Norstedts Juridik, Stockholm, 1992)

[186] Trevor Cox, *Chaos versus uniformity: the divergent views of software in the International Community*, 4 VINDOBONA JOURNAL OF INTERNATIONAL COMMERCIAL LAW AND ARBITRATION 3 (2000).

[187] Frank Diedrich, *The CISG and Computer Software Revisited*, 6 VINDIBONA JOURNAL SUPPLEMENT 55, 64 (2002).

is incorporated into the contract depends on successful communications of offer and acceptance between the parties. In the case of ordering software over the phone or otherwise in a "shrinkwrapped" form, such order may constitute communications of the parties preceding the presentation of the license. Thus, the license can be treated as a counter-offer (CISG Art. 19 (1)). Since the license terms would materially change the offer, it is not possible to apply the "last shot" rule of CISG Art. 19(3). In other cases, such as "click-wrap" transactions the license would usually form a straightforward offer. In either case, an offer or a counter-offer is effective only if it reaches the offeree (CISG Art. 15). Still, the contract is concluded only after the acceptance of an offer becomes effective (CISG Art. 23).[188] As in the practice of Open Source transactions programs are often simply downloaded from the Internet without communicating acceptance to the license terms, the basis for a valid contract can be found only in CISG Art. 18(3) which allows for indication of assent by performing an act. This may be helpful for Open Source licenses, which purport to treat the use of software as the assent to their terms. All then depends on the actual effectiveness of the communication of the license, in the language of CISG, whether it has reached the offeree.

One more important consequence of the applicability of CISG is the seller's obligation under Art. 42 (1) to deliver goods free from any right or claim of a third party based on industrial property or other intellectual property. By the virtue of Art. 42 (2), this obligation does not extend to cases where at the time of the conclusion of the contract the buyer knew or could not have been unaware of the right or claim. According to Diedrich, if only the seller is given a proper notice it is possible for the goods to be delivered together

[188] There is a CISG-Advisory Council Opinion on Electronic Communications under CISG (CISG-AC, *Opinion no. 1, Electronic Communications under CISG*, 15 August 2003, available at: http://www.cisg.law.pace.edu/cisg/CISG-AC-op1.html), but it is of little guidance for Open Source as well as other mass-market software developers, because it mostly concerns the email exchange of parties' communications.

with intellectual property rights limitations, such as the license.[189] In fact there are Open Source licenses which expressly address third-party claims to software, such as MPL and SISSL, but at the same time they exclude the applicability of CISG. The Apache License contains the disclaimer of warranty which excludes *inter alia* warranty of title. GNU GPL and BSD License do not specifically refer to third-party claims or title warranties, but they purport to exclude any express or implied warranties.

In the U.S., many courts that deal with "-wrap" software licenses have strongly relied on the UCC Art. 2 "Sales of goods". Pursuant to UCC 2-102, the whole Art. 2 applies to "transactions in goods", which are defined by UCC 2-105 as "movables"; "things in action" are at the same time expressly excluded. Furthermore, according to UCC 2-106 "sale consists in the passing of title from the seller to the buyer for a price". These definitions are the exact opposite of software licenses. It is possible; however, to treat software transactions as mixed contracts involving both goods and services and resort to various tests developed by the judiciary, which base the applicability of the UCC Art. 2 on the predominant purpose of transaction or the totality of circumstances.[190] Some courts did not enter into such sophisticated reasoning and simply assumed the applicability of UCC Art. 2 without providing any particular rationale. This was the case in such important decisions as *Step-Saver* or *ProCD*, whereas in *I.LAN Systems, Inc. v. Netscout Service Level Corp.*[191] the court noted that software licenses exist in a "legislative void" and assumed that UCC Art. 2 governs them only "for the time being".[192]

Some courts looked at software transactions more closely and ended up with not

[189] Diedrich, FN 187 at 70.
[190] See generally RAYMOND T. NIMMER, THE LAW OF COMPUTER TECHNOLOGY Sec. 6.01 (West Group, rev. ed. 1999).
[191] 183 F.Supp.2d 328 (D. Mass. 2002).
[192] *Id.* at 332.

applying UCC Art. 2. This happened in cases such as *Berthold Types Ltd. v. Adobe Systems, Inc.*,[193] *Adobe Systems Inc. v. One Stop Micro, Inc.*[194] or *Architectonics, Inc. v. Control Systems, Inc.*[195] In *Berthold*, the court relied on the UCC 2-106 definition of "sale" requiring the title to be transferred, which was not found in the license. In *Adobe*, the numerous restrictions found in the agreement were treated by the court as the proof of parties' intent not to sell software but merely to license it, even if they used such words as "purchase", "sell" or "buy". In *Architectonics*, the predominant feature of the agreement was found by the court to transfer intellectual property rights, thus not subject to UCC Art. 2. The rejection by the court of UCC Art. 2 leaves it with the common law of contracts, which seem not to be the best result for legal certainty, as it does not provide for a uniform and comprehensive set of gap-filling rules and above all is not tailored for contracting in the electronic age.[196] Had UCITA been enacted in the majority of states, it would preempt common law contracts rules for software licenses, but it is still not the case. The benefit of common law though is that "-wrap" contracts should generally be upheld under the common law "mirror image rule".[197] As a result of this rule, the terms of the party who sends the last form (*i.e.* "the last shot") govern the contract. A license found by the buyer after opening the box or appearing on the screen during installation is undoubtedly "the last shot".

4.1.4 Implications for Open Source Licensing

Open Source licenses have many features of ordinary mass-market software licenses. Namely, they are contracts of adhesion, with their terms fixed, giving the user no possibility

[193] 101 F. Supp.2d 697 (E.D. Ill 2000).
[194] 84 F. Supp.2d 1086 (N.D. Cal. 2000).
[195] 935 F. Supp. 425 (S.D. N.Y. 1996).
[196] Lorin Brennan, *Why Article 2 Cannot Apply to Software Transactions*, 38 Duquesne Law Review 459, 542 (2000).
[197] *Id.* at 543.

of negotiation. They resemble shrink-wrap and click-wrap licenses just discussed; however, rarely assent to their terms is linked to opening the box or clicking on "I agree" button. The common practice used by Open Source distributors is to attribute assent with some action of the user, like using the program, copying, modifying or redistributing it. Therefore, it is proposed to call such licenses "use-based".[198] It is very similar to the factual pattern which allowed the court not to enforce the software license in *Specht*.

GNU GPL seems to treat making the use of the program as the assent. It states in Sec. 5: "[B]y modifying or distributing the Program ... you indicate your acceptance of this License to do so, and all its terms and conditions." The "How to Apply These Terms to Your New Programs" explanatory section of GNU GPL requires to attach notices of license within the source code and, if the program is interactive, to make it output such notice when it starts in an interactive mode. The Apache License, BSD License, MPL and SISSL in sections covering notices all require to reproduce the license in the documentation, not in the source code of the program only. All the licenses are silent on how to provide the notice of the license if only the object code is distributed (except for GNU GPL's requirement to interactively output it).

Unfortunately, following any of this advice would make the license neither "shrink-" nor "click-wrap" and the only authority dealing with roughly the same factual pattern is *Microstar*, which did not invalidate the license but limited itself to a kind remark that the licensor could "do a better job" in making the user aware of the terms. Notices that the user may not even spot if the program is not interactive do not provide any opportunity to review the license. It means that the user could not be aware that the license treats his certain

[198] Stephen T. Keohane, *Mass Market Licensing*, in: 652 Practising Law Institute, Patent & Technology Licensing 2002 269, 278 (2002).

conducts as assent to the terms, which would put its validity into question under both UCC and CISG. Under European Law there are also doubts whether the commercial communication could be regarded as clearly identifiable if it is provided in such a way.[199]

At the first glance, it seems that there would be no valid communication between parties; however, it does not have to be so in every case. One helpful fact is that although there are many Open Source licenses, they come in the form of widely known model standard form contracts. Contrary to proprietary software vendors, who draft separate licenses for every piece of software, Open Source developers often decide to use one of model licenses, such as GNU GPL. As a consequence, because these model licenses are known or at least easily accessible to the general public, in a particular dispute the difficulty may not lie in proving that the licensee knew the license, but just the fact that the program was subject to it.

Certainly, the knowledge about the license not expressly presented to the user always remains subject to proof. Therefore, the only solution to secure greater probability of validity and enforcement of Open Source licenses is to make their contracting procedure similar to proprietary licensing and provide for clear communications between the parties, such as clicking on "I agree" button as a necessary condition to download or use the program.[200]

4.2 Copyright Law and Open Source Licensing

Whichever contract law regime is found to apply to a software transaction, what

[199] The conformity with obligations to provide consumers with adequate prior information should be evaluated for each Open Source project separately. Generally speaking, commercial Open Source developers comply with them better than non-profit coordinators. Non-profit character, however, does not exclude suppliers from the scope of consumer protection directives. In the German case discussed before, the court found that GNU GPL formed the parties' contract despite the lack of "shrink-wrap" or "click-wrap" method, but neither party was a consumer in that case.

[200] Christian H. Nadan, *Open Source Licensing: Virus or Virtue?* 10 TEXAS INTELLECTUAL PROPERTY LAW JOURNAL 349, 367 (2002).

could have the greater importance is its relationship with copyright laws' provisions and the effect it has on the license recognition. As it was already discussed, intellectual property laws set a scope of private exclusive rights and balance them with the access right of the public. Whether such balancing provisions are mandatory or simply set a presumption or default rule is of enormous importance for concluding whether the parties are allowed to contract in other way. As intellectual property laws are not always straightforward, there is a number of legal doctrines which address this issue and they vary with regard to different national jurisdictions.[201] The U.S. sources of limits set by federal and state intellectual property laws on the ability to change the licensing rules by contract, which seem especially relevant in the discussion on the validity and enforceability of Open Source licenses, are: (1) preemption and (2) copyright misuse.[202] In Europe, the Continental copyright law institute of (3) moral rights seems to be the most relevant for the purpose of this paper.

4.2.1 Preemption

In the U.S., copyright and contract law relationship is influenced by the federal structure of the law. There is one federal Copyright Act, whereas each state has its own contract law, consisting of common law and the particular implementation of UCC.[203] To put it in a nutshell, state laws have to conform to federal laws and if there is no such conformity, federal laws preempt them. There are two bases of preemption relevant for the subject of this paper: Copyright Act Sec. 301 and the Supremacy Clause. Federal Copyright Act preempts

all legal or equitable rights that are equivalent to any of the exclusive rights within

[201] An example of relatively clear-cut provisions may be found in European Software Directive Art. 5, which expressly states that the rights it grants to the lawful acquirer may be contracted away and at the same time provides for certain rights which cannot be prevented by contract.

[202] Mark A. Lemley, *Beyond Preemption: The Law and Policy of Intellectual Property Licensing*, 87 California Law Review 111, 113 (1999).

[203] For a thorough analysis of incompatibilities between UCC Art. 2 and Copyright Act see Brennan, FN 196 at 481 *et seq*.

> the general scope of copyright ... and come within the subject matter of copyright.[204]

It means that states cannot regulate the rights, which are already covered by the federal Copyright Act. The Supremacy Clause of the U.S. Constitution states:

> This Constitution, and the Laws of the United States which shall be made in Pursuance thereof, ... shall be the supreme Law of the Land; and the Judges in every State shall be bound thereby, any Thing in the Constitution or Laws of any State to the Contrary notwithstanding.[205]

In other words, state law cannot interfere with federal policy and "stand as an obstacle to the accomplishment and execution of the full purposes and objectives of Congress."[206]

Thus, the court presented with the question of the validity and enforceability of a software license must perform a twofold analysis. Pursuant to Copyright Act Sec. 301 it must ask whether the state rights sought to be enforced are equivalent to rights established by this Act. Furthermore, it must inquire, whether such enforcement would not interfere with the policy decisions underlying the Act. This is not always easy to distinguish and some arguments may seem equally supportive in either stage.[207] The point which undoubtedly falls under Copyright Act Sec. 301 is that copyrights are rights against the world (rights *in rem*), whereas contracts may generally create rights only between their parties (rights *in personam*) but not exclusive rights that could be enforced against third persons. Such line of reasoning allowed Judge Easterbrook not to preempt "shrink-wrap" license in *ProCD*. The practical

[204] Copyright Act, 17 U.S.C. Sec. 301(a) (1976, as amended).

[205] U.S. Const., Art. VI, Sec. 2.

[206] *Hines v. Davidowitz*, 312 U.S. 52, 67 (1941).

[207] For example, according to Karjala, the lack of actual bargain in adhesive "shrink-wrap" licenses leaves no state interest in enforcing a claim resulting from such contract, which in reality is no different than a copyright infringement claim; whereas for Founds this is a good argument only for preemption under Supremacy Clause. Compare: Dennis S. Karjala, *Federal Preemption of Shrinkwrap and On-Line Licenses*, 22 University of Dayton Law Review 511, 527-528 (1997) with Garry L. Founds, *Shrinkwrap and Clickwrap Agreements: 2B or not 2B?*, 52 Federal Communications Law Journal 99, 109 (1999).

effect of software licensing; however, does not fall neatly under the theoretical *in rem – in personam* distinction. To some extent, licenses indeed attempt to create rights against the world, as they claim to bind any person using the software. Users cannot bargain for the terms and the only way for them to access the program is to accept the license. Indeed, by accepting it they become parties to the contract and subject themselves to *in personam* claims, but the only other alternative is to become copyright infringers. Such reasoning is explicit in GNU GPL Sec. 5, which states:

> You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License.

According to Karjala, preemption does not influence terms of warranties, rights of return or "similar economic aspects of the transaction", but attempts to extend the contractual relationship to third parties would be preempted if as the result of such extension rights similar to copyrights were created.[208] Nimmer looks at the case law and concludes that the success of Copyright Act Sec. 301 preemption claim depends much on whether the software license was enforceable pursuant to general state law and that successful preemption claims are limited to cases where there was no contractual or otherwise special relationship between the parties.[209] Although GNU GPL attempts to bind everyone performing actions described in it, it includes a clause that may be regarded as the basis of such special relationship with these third-persons. GNU GPL Sec. 6 states:

[208] Karjala, FN 207 at 529.
[209] Raymond T. Nimmer, *Breaking Barriers: The Relations Between Contract and Intellectual Property Law*, 13 Berkeley Technology Law Journal 827, 862 (1998).

> Each time you redistribute the Program ... the recipient automatically receives a license from the original licensor ... subject to these terms and conditions.

This clause is a rather straightforward attempt of the copyright owner to conclude contracts with all the subsequent recipients. Whether it indeed creates a contract valid under state law is the matter of contract formation rules discussed above, which makes the Copyright Act Sec. 301 preemption analysis dependent indirectly on giving the proper notice about license terms.

It remains to be decided upon, whether the licensor's rights provided for in an Open Source license are greater than those described in Copyright Act Sec. 106. The relevant rights are: reproduction right, distribution right and the right to prepare derivative works. GNU GPL grants all these rights and provides additionally in Sec. 2(b):

> You must cause any work that you distribute or publish, that ... contains or is derived from the Program ..., to be licensed as a whole at no charge to all third parties under the terms of this License.

According to Gomulkiewicz, GNU GPL attempts to define "derivative works" wider as it is provided in Copyright Act Sec. 101 and may purport to cover works derived from such uncopyrightable elements as ideas or data.[210] Even if such broad interpretation of the license seems rather ungrounded, it is indeed true that there are not enough guidelines when exactly the licensee's program would be subject to "copyleft virus".[211] On the other hand, the programming techniques constantly develop and allow for designing various complicated

[210] Gomulkiewicz, FN 116 at 90. Compare Copyright Act Sec. 101 "A derivative work is a work based upon one or more preexisting works, such as a translation, ... abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations, elaborations, or other modifications, which, as a whole, represent an original work of authorship, is a derivative work." with GNU GPL Sec. 0 defining "work based on the Program" as "either the Program or any derivative work under copyright law."

[211] For example, GNU GPL Sec. 2 *in fine* allows for distribution of independent and separate works that are not derived from a program under a different license, but at the same time subjects them to "copyleft" if they are distributed "as part of a whole which is a work based on the Program."

relationships between programs. Thus, were the “copyleft” clause drafted more specifically it could not serve its purpose too long. This vagueness is remedied to some extent by FSF, which publishes GNU GPL FAQ, where it explains the applicability of “copyleft” to, for example, the program's output, linked program, main program to which the free program is a “plug-in” *etc*.[212] Usefulness of these explanations depends on the circumstances of particular case.

In order to be valid under Copyright Act Sec. 301 preemption analysis, GNU GPL Sec. 2 should be interpreted to cover derivative works within the meaning of Copyright Act Sec. 101 only, otherwise it would indeed broaden the licensor's rights beyond their scope described in Copyright Act Sec. 106. The real problem; however, is to determine what constitutes a derivative of a program. According to Larry Rosen merely using a program licensed under GNU GPL or even combining it with the licensee's own program by linking them dynamically or making them interact using application program interface (API) does not involve the creation of derivative works; thus, the “copyleft” would apply only to statically linked programs.[213] On the other hand, GNU GPL FAQ, which considers many more ways of making programs interact together, lists dynamic linking as covered by “copyleft”. Untangling this dispute undoubtedly requires not only the sophisticated technical expertize but also the knowledge of specific understanding of “derivative works” in particular jurisdiction. Thus, it seems impossible to give a general answer, although in a given case “the fear that the licensee would have to publish his own source code [might be] exaggerated.”[214]

GNU GPL is not the only license with “copyleft” clause and the developers using

[212] Free Software Foundation, *Frequently Asked Questions about the GNU GPL*, http://www.fsf.org/licenses/gpl-faq.html.
[213] Larry Rosen, *The Unreasonable Fear of Infection*, 2, available at: http://rosenlaw.com/html/GPL.PDF.
[214] *Id.* at 3.

Open Source programs covered by other licenses also have to pay close attention to its relationship with their intellectual property. Apache License contains its own express definition of "derivative works", which partially relies on the language of Copyright Act Sec. 101, but it is not its verbatim copy.[215] BSD License does not contain a "copyleft" clause, but also does not expressly allow for the creation of derivative works; it merely grants the right to redistribute modifications. Similarly, such "copyleft" licenses as MPL and SISSL do not grant the right to prepare "derivative works" but only "modifications".[216] Generally, these clauses are less vague than the "copyleft" of GNU GPL; thus, their scope should be easier to determine in a particular case and, more importantly, the copyright preemption argument against them is weaker.

After Copyright Act Sec. 301, there comes Supremacy Clause preemption analysis, which involves policy considerations. As it was already presented, the policy underlying the U.S. Copyright Act, set forth in the Intellectual Property Clause is the incentive theory. In order to balance proprietary control with public access, the Congress limited broad exclusive rights of copyright owners with such limitations as idea-expression dichotomy, first sale and fair use doctrine. They are there to ensure the transfer of knowledge to the public while still securing enough private interest of the authors. The question then is whether Open Source licenses retain this carefully designed balance. What can be noticed at the first glance is that

[215] Apache License Sec. 1 "Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

[216] MPL Sec. 1.9. "Modifications" means any addition to or deletion from the substance or structure of either the Original Code or any previous Modifications. When Covered Code is released as a series of files, a Modification is:
A. Any addition to or deletion from the contents of a file containing Original Code or previous Modifications.
B. Any new file that contains any part of the Original Code or previous Modifications."
SISSL contains a very similar definition of "modifications".

all the provisions of proprietary software licenses that are usually mentioned as susceptible of Supremacy Clause preemption are not present in Open Source licenses. Namely, instead of retaining control over the program, they encourage its reproduction and redistribution. Moreover, they do not prevent users from decompilation[217] but allow for exploring the software and for modifications to a far greater extent that the default Copyright Act provisions on fair use. Whereas proprietary licenses tend to tilt intellectual property balance towards the private owners, the "problem" with Open Source licensing may be that it reaches for the other extreme.

Indeed, the critics of Open Source ideology often repeat that it provides no incentives to write software, as the authors retain no control over its substance, nor can they secure profits from distribution. The straightforward answer to this may be that no-one forces the hackers to share their software with the world. Thus, the mere fact of the existence of Open Source phenomenon proves that there exist enough incentives to supply software in this production scheme. The nature of these incentives seems to be more complex than the simple and prosaic monetary award of proprietary model and there is an ongoing debate what actually they are.[218] One often repeated hypothesis is that the Open Source Community is based on "gift culture", which appears in societies having surpluses of goods and rewards those who share with others. Briefly speaking, hackers seem to find incentives in their recognition as the most productive contributors, not in direct monetary award. Because there is no textual basis in the Intellectual Property Clause to support the argument that the incentive theory regards monetary incentives as the only possible way "to promote the Progress", Open Source licensing model does not seem to stand against the underlying policy

[217] The availability of source codes limits the need of decompilation significantly.
[218] For a thorough analysis see various papers by Eric S. Raymond.

of Copyright Act as defined by the U.S. Constitution.

The argument may be raised that the decision to license software as Open Source can be the result of some external influence on the developer which forces him to forgo his private interests otherwise secured by Copyright Act, especially in the light of so called “viral effect” of the GNU GPL “copyleft” clause. As it was already presented, “copyleft” requires the licensee to distribute all his derivative works of the program under the same license. This means that they have to be made freely available to anyone for reproduction and further modification together with their source codes. Again, a lot depends on the scope of Open Source licensing and especially its understanding of “derivative works”, “works based on the program” or any other term used by the Open Source licenses to set the scope of copyleft.

It can be concluded that the enforceability of “copyleft virus” might be preempted pursuant to Supremacy Clause, only to the extent that it would attempt to open the source of software, which is unrelated or developed independently from the original program. This conclusion is similar to the one reached in Copyright Act Sec. 301 preemption analysis presented above. However, the effect here seems to depend not on the scope of rights that an Open Source license is trying to enforce, but much more on the fact that the software becomes subject to “copyleft virus” only by its author's deliberate decision to both modify and redistribute the original program and that the licenses do not attempt to make him dispose of his legally acquired copyrights. Indeed, Open Source licensing is different than releasing software into public domain. GNU GPL expressly states in its preamble “(1) [we] copyright the software, and (2) offer you this license”; thus, all rights granted by the Copyright Act remain with the author, who licenses them as the developers of proprietary software do. As the result, not only the protection of Copyright Act is not denied by Open Source licensing, but also it provides for a mechanism of allowing the public access to the software subsequently developed after the initial release of Open Source program. Rosen explains the "copyleft" bargain in the following way: "You can create derivative works of GPL-licensed programs only if you agree to return the favor."[219] Thus, the effect of "copyleft" is exactly what the intellectual property laws' underlying policy aims at – to secure supply of knowledge and innovation for the society. If Open Source proves, as it in fact does, to provide enough incentives for the authors to create such supply, one may risk a conclusion that it Solomonically resolves the old conflict between private and public interests in the knowledge market, which is what the Intellectual Property Clause was drafted for.

4.2.2 Copyright Misuse

Similarly to preemption, copyright misuse doctrine may result in not enforcing a software license.[220] This doctrine, deriving from an older patent misuse doctrine, prohibits enforcing a copyright broadened beyond its statutory limits by such conduct of its holder as certain licensing practices.[221] It is not necessary for the court to find antitrust liability in order not to enforce a misused copyright and it is also not necessary for a person raising misuse defense to be actually harmed or affected by it.[222] However, the most important limit of copyright misuse is that it can only be used as a defense against copyright infringement claims, thus it may not help against breach of contract claim in the same case.[223] The doctrine was applied in *Lasercomb America, Inc. v. Reynolds*,[224] where the court did not allow to enforce a standard form software license containing a clause prohibiting the development of

[219] Rosen, FN 213 at 3.
[220] Lemley, FN 202 at 144-152.
[221] 2 Paul Goldstein, Copyright, 9:35-40 (Aspen Law & Business, New York, 2nd ed. 1998).
[222] Nadan, FN 200 at 368.
[223] Lemley, FN 202 at 157.
[224] 911 F.2d 970 (4th Cir. 1990).

any competing software for a 99-year period.[225] It is clear that the holding in *Lasercomb* should not worry Open Source licensors, as they expressly permit the use of their software in any manner, even in order to create a competing product.

However, the underlying idea of misuse doctrine should be analyzed. In the cases preceding *Lasercomb*, the courts examined patent grantback clauses, which usually require to license back to the licensor any improvements or derivatives that were created by the licensee on the basis of the licensed original. Patent misuse was found in requiring the licensee to grant back preexisting or unrelated patents. Nadan, who discusses the misuse defense in more detail, presents this as yet one more threat to the enforceability of GNU GPL "copyleft" clause, which "purports to affect independent, separate works."[226] Whether this is really what "copyleft" clause aims at is not so apparent, especially in the light of the analysis contained in previous section, but it is necessary to stress again that its exact scope is crucial for the success of Open Souce licensing. Having in mind the fact that "copyleft" is practically the only restriction for licensees and that it is placed in the most popular Open Source licenses, whether it stands against threats such as misuse defense may prove important for the whole future development of this software production model.

In any case either the scope of "copyleft" clause or the meaning of a derivative work of a computer program needs clarification. It is not easy to determine what should it be in the complex reality of software business, where programs interact in a variety of ways and new programming techniques are constantly being developed. At the roots of "copyleft" licensing rests the idea similar to the one in patent grantback clauses. The creation of improvements by the licensee is possible only because of the access to the original licensed intellectual

[225] *Id.* at 978.
[226] Nadan, FN 200 at 369.

property; thus, to require him to contribute them back to certain extent seems a fair trade. What is extremely important with "copyleft", is that it demands for the contribution to the general public, but at the same time it grants access to the original to everybody and for free. It must be also stressed again that the obligation to contribute is triggered only by the licensee's independent decision both to modify and redistribute the software. Because of the cooperativeness of Open Source development, the compliance of the parties with the license affects the whole society benefiting from new software supply or at least the whole Open Source Community. All of these special conditions should be taken in mind when deciding whether the misuse defense should allow not to enforce "copyleft". According to Potter, "[copyleft] is minuscule when compared to the restrictions found in proprietary licenses"[227], such as those prohibiting backup copies or reverse engineering and allowing to run the program on one computer only. Also in the mind of the author of this paper, Open Source licenses definitely call for more tolerance than the one given by the courts towards grantback clauses.

Patent and copyright misuse cases usually involve competition law issues. In Europe, where there is no Community-wide intellectual property misuse doctrine, the relevant law could be the Commission Regulation on the application of Article 81(3) of the Treaty to categories of technology transfer agreements (Technology Transfer Regulation).[228] It defines technology transfer agreements to encompass software licenses and generally declares Article 81(1) of the Treaty not applicable to them. However, apart from prohibiting some hardcore restrictions in Art. 4, the Art. 5 of the regulation contains express prohibition of exclusive grantback clauses. *A contrario*, obligations of the licensee to grant a non-exclusive license to

[227] Shawn W. Potter, *Opening Up to Open Source*, 6 Richmond Journal of Law and Technology 24, 74 (2000).
[228] 2004/772/EC, OJ L 123, 27/04/2004, P. 11.

the licensor or to a third party in respect of its own improvements should be treated as covered by the exemption. Moreover, since “copyleft” clauses increase the ratio of technology transfer because the obligation to share one's improvements is automatically triggered upon distribution and thus they give all competitors access to the same technology, the argument of their anti-competitive effect seems very weak.

4.2.3 Moral Rights

Moral rights protect the relationship between the author and his work and for example grant him the right to have his authorship recognized or they protect the integrity of the work by allowing the author to object to its derogatory treatment. Open Source Software development strongly relies on the reputation gain of the hackers and possibility of a wrongful attribution of subsequent program versions is even expressly addressed by some of the Open Source licenses requiring to properly preserve copyright notices and indicate which parts of the program were changed by whom. For that, additional protection may be sought in moral rights, which are particularly strong in countries following the Continental approach towards intellectual property law. The worldwide statutory basis for this type of Open Source Software protection may be found in Art. 6bis of the Berne Convention, which applies to computer programs pursuant to such instruments as WIPO Copyright Treaty or European Software Directive.

Metzger and Jaeger note that the very broad grant of rights in GNU GPL does not completely exclude the author's right to protect the integrity of the work covered in Sec. 14 of the German Copyright Act.[229] They conclude that it is not possible to grant the right to modify

[229] Axel Metzger, Till Jaeger, *Open Source Software and German Copyright Law*, 32 International Review of Industrial Property 52, 65 (2001).

the work “to the extent of distortion.”[230] It means that the initial author is empowered with some residual control over the Open Source Software and provides an example how statutory law can strengthen moral norms which already recognize “project coordinators” as the only persons allowed to decide on the project's development or perceive “forking” as the last resort if the coordinator undertakes objectively bad decisions.

Therefore, contrary to preemption and copyright misuse doctrines analyzed above, moral rights should not be perceived as a way of invalidating an Open Source license but as an additional and subsidiary institute securing the proper development of software. They do not allow to interpret the broad grant of rights in the licenses as a complete control waiver and prevent licensees from taking away the community's most valuable assets – their source codes and their reputations.

4.3 Possible Consequences of License Invalidity

Should a software license not stand the court's scrutiny under either of the legal theories presented above, there is a need to determine the valid basis for the end-user's rights to use the software that he legally purchased or otherwise acquired in good faith. Indeed, intellectual property laws contain rules which are helpful in this second stage of analysis. There are three bases of users' rights which are relevant in the discussion of Open Source Software licenses: (1) first sale; (2) fair use; and (3) implied licenses, but they do not need to be extensively addressed, because the invalidity of a whole license is rather a theoretical possibility only and on the example of the U.S. law it will be shown that neither party of Open Source Software transaction would likely opt for such solution. However, it is possible that the invalidity of only particular clauses of licenses is invoked, the most crucial of which

[230] *Id.* at 66.

are warranty disclaimers and liability limitations. In such a case it is necessary to determine (4) which liability regime would govern the users' rights in case the software proves defective.

4.3.1 First Sale

In the U.S., pursuant to Copyright Act Sec. 109

> the owner of a particular copy ... lawfully made under [Copyright Act], ... is entitled, without the authority of the copyright owner, to sell or otherwise dispose of the possession of that copy.[231]

This provision sets an exclusive right exhaustion rule known as "first sale". Some of the American courts, usually after finding that UCC Art. 2 applies to the software transactions at issue and treating the license as a sale, logically turned towards the first sale doctrine. This is not an automatic result and in cases such as *ProCD* or *MA Mortenson* the courts ended up with enforcing the license despite finding UCC Art. 2 applicable.

In at least two cases; however, the first sale doctrine allowed the courts not to enforce the license accompanying the transaction. In *Novell, Inc. v. Network Trade Center, Inc.*,[232] the "shrink-wrap" license was held void, to the extent it attempted to retain title in the seller. Similarly, in *Softman Products Co., LLC v. Adobe Systems, Inc.*,[233] the fact that the software contained a "click-wrap" End-User License Agreement (EULA) was not enough to construct the transaction as license. Softman was unbundling and redistributing the programs legally acquired on the market as "software collections", but never installed them, thus was not made aware of the license clause prohibiting such unbundling. The license failed contract formation analysis for lack of appropriate notice and no expression of assent. The court

[231] Copyright Act, 17 U.S.C. Sec. 109 (a) (1976, as amended).
[232] 25 F. Supp. 2d 1218 (D. Utah 1997).
[233] 171 F. Supp. 2d 1075 (C.D. Cal. 2001).

backed it up with the first sale doctrine and held that “a single payment for a perpetual transfer of possession is, in reality, a sale of personal property and therefore transfers ownership of that property, the copy of the software”.[234]

The results of *Novell* or *Softman* seem not to be the most dangerous threat for Open Source licensing. First of all they are not fully theoretically correct in distinguishing between the copy of a program and the copyrighted program itself, to which the first sale limitation does not apply. In other words, although the legal purchaser of a “shrink-wrapped” program can legally dispose of such copy, that does not give him any intellectual property rights in this program. Furthermore, Open Source licenses do not prohibit, but expressly allow redistribution, so there is not much difference in practice whether redistribution right would flow from the license or first sale. As the first sale allows the purchaser only to sell or otherwise lawfully dispose of the copy, there is no danger that it could be used to undertake an action that Open Source licenses prohibit.

4.3.2 Fair Use

The German decision serves as an example that what really may threaten Open Source licensing is not just redistribution but additionally the closing of the code of Open Source programs. Open Source license infringers would like to upgrade the software by some modifications, not to share the new code with the general public and seek for rents flowing from the resulting monopoly. This cannot be done with the use of the first sale doctrine, as it allows to redistribute the copy as it is only. Moreover, this is what Open Source licenses and especially GNU GPL expressly prohibit. Once the licensee decides to distribute modified software legally, the source code must be made available also.

[234] *Id.* at 1086.

The question then is, whether the provisions of fair use would allow to circumvent the licenses and to close the program's code. Again, the logic of GNU GPL Sec. 5 seems to prevail even if the license itself would be held invalid. Copying, creating derivative works and other actions that would have to be undertaken in order to free-ride on Open Source Movement are expressly prohibited by law. The fair use provision in Copyright Act Sec. 107 is too limited to allow this, especially if the free-rider made commercial use affecting the potential market for the program. As the court held in *Sony Corp. of Am. v. Universal City Studios, Inc.*,[235] "every commercial use of copyrighted material is presumptively an unfair exploitation of the monopoly privilege that belongs to the owner of the copyright." Thus, either the free-rider limits himself to small portions of code and non-commercial use or he decides not to question the validity of the Open Source license that extends rights allowed by fair use provisions and becomes the part of the community.

4.3.3 Implied Licenses

In the case of both Open Source and proprietary programs distributed via Internet, they are often freely and publicly available with no express or easily visible terms attached. As the use of the program without authorization usually constitutes copyright infringement, persons acquiring it over Internet could easily be induced to infringe if there was no legal protection of their good faith actions. Fair use may allow them legally use programs distributed in such way but as Kosturakis points out, if there are no reservations accompanying the transaction, the user's rights should be construed according to an implied license, the scope of which can be determined by particular circumstances or the industry's custom.[236]

[235] 464 U.S. 417, 451 (1984).

[236] Irene Kosturakis, *Software Licensing and UCITA*, in: 762 Practising Law Institute, Understanding the

Before considering the existence of implied contract, one should note that simple putting a program on the Internet for free downloading and using might constitute a waiver of copyright, thus making it "public domain software". Such software can be used and modified without risking copyright infringement. However, the author of the upgraded program, which is the result of such modification, would often be protected by copyright. Consequently, he would be able to control the distribution and use of the upgrade. If the courts, after finding no valid Open Source license conclude that the developer actually waived his copyright, they would allow persons who obtained the source code of original program to upgrade it and use copyrights in the upgrade to close the code. Even if the original remained freely accessible, obviously it is the upgrade which would gain the market, where customers at all times demand more advanced programs. Fortunately, Open Source developers may rely on the holding in *Storm Impact, Inc. v. Software of the Month Club*,[237] which considered the distribution of the program by placing it for a free download on the Internet. According to the court, such circumstances do not constitute a copyright waiver or give rise to the conclusion that the author allowed for unlimited distribution.[238] Also, the *dicta* in *Planetary Motion*[239] must be recalled, where releasing the program under GNU GPL was enough for the court not to find a waiver. Indeed, the preamble of GNU GPL expressly states: "we copyright the program". Thus, it seems that only a complete layman would equate Open Source with "public domain software"; Open Source Movement does not need to fear that the closing of the code could legally occur in this way.

There remains the possibility that the copyrights to an Open Source program are not

INTELLECTUAL PROPERTY LICENSE 2003 437, 448-449 (2004).

[237] 13 F.Supp. 2d 782 (N.D. Ill. 1998).

[238] *Id.* at 791.

[239] FN 159.

regarded as waived but the license after the scrutiny of contract formation rules is rendered invalid. Then, the user might want to rely on an implied license, allowing him to use the program for the purpose determined by the circumstances of the transaction. According to Nadan, it is clear that Open Source Software is "provided for the express purpose of being modified, used and distributed;" thus, should the court not find an express license it would probably allow to use, modify and redistribute the program but it would rather not find such unusual clause as "copyleft" to be impliedly agreed upon.[240] It means that the proprietarization of the program might be legally possible. On the other hand, it is very risky for a user to rely on the existence of only implied license for redistributing and modifying. If the court subsequently finds out that there was no such contract in given particular circumstances it would subject the user to copyright infringement claims or maybe even criminal proceedings. Simply admitting the knowledge and assent to an Open Source license gives modification and redistribution rights expressly to the user and without any risk.

Moglen points out that there is no need of a license for a lawful acquirer of a program to use it or even experimentally modify it – these rights are already provided for in copyright statutes either impliedly within fair use or expressly in the provisions covering copying in the course of normal use of the program or allowing for limited decompilation.[241] Gomulkiewicz requires however, that the right to use the program is expressly provided in the license and bases this argument on the premise that the use of a work outside the scope of a license is a breach of contract and a violation of Copyright Act.[242] There is no express permission of use in GNU GPL and Apache License,[243] but BSD License, MPL and SISSL

[240] Nadan, FN 200 at 365-366.
[241] Moglen, FN 121.
[242] Gomulkiewicz, FN 116 at 84-85.
[243] The right of use is granted only in the Sec. 3, covering patent license. At the same time Apache Foundation admits that they do not hold any patents.

grant such. Indeed, the users may expect all their rights to be covered by the license, but it is common knowledge that default statutory provisions govern every contract even if not expressly invoked; moreover, the right to use is not one of the exclusive rights of the copyright holder. Thus, Gomulkiewicz's arguments should not be treated as a serious threat to Open Source licensing and clearly Moglen is much more convincing here.

Gomulkiewicz makes a point; however, about BSD License, which does not contain an express clause allowing for modifying the program.[244] Therefore, the right to create modifications has to be interpreted from the grant of the right to redistribute modifications expressly included in this license. As it would be pointless to allow the user to redistribute modifications without allowing him to create them in the first place, the grant of the latter right is logically covered by BSD License.

4.3.4 Liability for Open Source Software

There are different statutory liability regimes that could apply to software transactions should the limitation and disclaimer contained in the license turn out invalid. They can be found in any national contract law a court would find applicable in a given situation and they are also contained in CISG. For example, if the parties fail to communicate properly the terms of license or the assent to them and the court finds some other basis for a valid contract, it could refer to CISG Art. 35, the relevant part of which provides that unless expressly agreed to the contrary, goods should be fit for the purposes for which goods of the same description would ordinarily be used or, in the case of buyer's reliance on the seller's skill and judgment, they should be fit for any particular purpose made known to the seller. Similarly, UCC 2-314 contains *inter alia* the requirement of the fitness for ordinary purposes

[244] Gomulkiewicz, FN 116 at 94.

and UCC 2-315 for any particular purpose.

These obligations are exactly what the Open Source licenses intend to exclude. In fact, all of the licenses discussed in this paper contain language which by the virtue of UCC 2-316 is sufficient to exclude all implied warranties (*i.e.* “as is” expressions). But it has to be stressed again that these disclaimers are effective only if the license terms become part of the parties' contract. The language seems at first glance to comply also with the requirements for disclaimers set in UCITA (Sec. 401 *et. seq.*) but again, everything depends on whether the license as a whole was validly concluded between parties.

Limitations and disclaimers are also under threat of consumer protection laws, which are particularly strong within the European Union. Many of these regulations concern liability for defective goods, as is the case with the Directive on certain aspects of the sale of consumer goods and associated guarantees[245] or the Directive on the approximation of the laws, regulations and administrative provisions of the Member States concerning liability for defective products.[246] The former defines consumer goods as “any tangible movable items”, the latter refers to “all movables”. It seems that their applicability to software cannot be definitely excluded but one strong argument against it flows from the need to preserve consistence of the whole body of European Union Law. Such consistence would be impaired if software was regarded as goods for the purposes of abovementioned directives and at the same time as falling under the definition of “information society services” and its suppliers required to comply with the E-commerce Directive.

Regardless of the status of software as goods or services, mass-market software licenses should be searched for any unfair contract terms. The European Directive on unfair

[245] 1999/44/EC, OJ L 171, 07/7/1999 P. 12.
[246] 85/374/EEC, OJ L 210, 07/08/1985 P. 29.

terms in consumer contracts (Unfair Terms Directive),[247] regards any contractual term which has not been individually negotiated as unfair if, contrary to the requirement of good faith, it causes a significant imbalance in the parties' rights and obligations arising under the contract, to the detriment of the consumer (Art. 3.1). The fact that the directive demands the interpretation most favorable to the consumer in case of doubt immediately brings to mind MPL and SISSL, which contain express clauses excluding the applicability of laws requiring construction against the drafter. These licenses also contain choice of law provisions, pointing at Californian Law, but Member States are expressly prohibited (Art. 6.2) to deprive the consumer of the protection by enforcing the choice of the law of a non-Member country as the law applicable to the contract if there is a close connection with the territory of the Member States. Similar mechanism exists in Art. 5 of the Convention on the Law Applicable to Contractual Obligations (Rome Convention). This provision has to be taken under consideration also in the case of these Open Source licenses that lack choice of law clauses, as it points to the law of the consumer's country in such situation (Rome Convention Art. 5.3). It can be concluded that the applicability of consumer protection laws of both the Community and its Member States cannot be excluded with regard to Open Source Software users.

It must be stressed again that liability limitations and warranty disclaimers are not Open Source-specific clauses. Proprietary software vendors use them too because their software also contains many untraceable bugs. Thus, all of them are interested in their enforceability. It is; however, by far more important for Open Source developers that these clauses prove valid because it would be hard to find any hacker willing to contribute to a

[247] 93/13/EEC, OJ L 095, 21/04/1993 P. 29.

project for free, knowing that he could be subject to liability.

A quick search of major Open Source licenses for the terms defined in the Annex to the Unfair Terms Directive leads to the conclusion that the drafters of the licenses should probably consider making them in line with at least items b) and i) thereto. Item b) prohibits inappropriate exclusion or limitation of the legal rights of the consumer in the event of total or partial non-performance or inadequate performance; thus, logically it is the relevant provision for the evaluation of liability limitations and warranty exclusions. A good argument here is that the provision of the software "as is", when distributed free of charge together with source codes, is clearly appropriate and fair. Item i) of the Annex does not allow to irrevocably bind the consumer to terms he had no real opportunity of becoming acquainted with before the conclusion of the contract. Thus, it is necessary to indicate once more that express and clear communication of Open Source license terms must be provided for.

One of the ideas developed in this paper is that Open Source Movement does not treat the users as mere consumers but by giving the broad rights and access to source codes equates them with developers. Theoretically, such arrangement provides no reason why users should feel dependent or inferior when entering into legal relations with suppliers. These facts support the argument that consumer protection laws aiming at securing the interests of the weaker party should not be applicable to Open Source transactions. However, many software end-users are acting outside their trade, business or profession, that makes them consumers, as defined by European and other consumer protection laws. Apart from this legal argument, the fact that users continue to depend on particular software providers (companies such as Red Hat and SuSE or non-profit distributors such as Debian), despite the possibility of maintaining software on their own or commissioning this to anybody, calls for retaining the higher level of their protection.

4.4 Implications of "Copyleft" Infringement

Assuming the validity of an Open Source license it must be considered whether various "copyleft" clauses could serve their purpose well and prevent the program from being proprietarized. Unless otherwise construed, "copyleft" clauses in their parts referring to the distribution of the program's modifications constitute the use of the exclusive author's right to authorize the creation of derivative works, which are subject to copyrights independent from the protection of the original. The protection of derivative works; however, "does not extend to any part of the work in which the [original material] has been used unlawfully," to take the U.S. Copyright Act Sec. 103 (a) as an example.

The author of the derivative work is entitled to distribute it, subject to the original author's consent. However, this rule applies only to lawfully created derivative works and if such work was created without authorization or outside the scope of fair use limitation, copyright law does not protect it. In the U.S. it is agreed that neither the infringer nor the original author can be considered to hold copyrights in unauthorized derivative works, but the jurisprudence fails to produce a positive answer on their legal status.[248] Lemley, for example, argues that they belong to *public domain*.[249] If this is the case, then it could constitute a loophole for making an Open Source program proprietary. There seems to be no threat if the

[248] Sean Hogle, *Unauthorized Derivative Source Code*, 18 COMPUTER AND INTERNET LAWYER 1, 5 (2001). In other jurisdictions the assignment of rights to program derivatives can be different. For example, according to Polish jurisprudence, the rights to modifications of computer program belong to the author of the original as a matter of law, except for the right to claim authorship. Compare: M. Byrska, *Prawne aspekty modyfikowania programu komputerowego [Legal aspects of modifying computer program]*, 4 KWARTALNIK PRAWA PRYWATNEGO [PRIVATE LAW QUARTERLY] 693 (1996) (applying this rule to any modifications and allowing their author to claim the moral right of authorship only) with: A. NOWICKA, PRAWNOAUTORSKA I PATENTOWA OCHRONA PROGRAMÓW KOMPUTEROWYCH [COPYRIGHT AND PATENT PROTECTION OF COMPUTER PROGRAMS], 134 (Dom Wydawniczy ABC, Warszawa, 1995) (applying this rule only to modifications made without the original author's consent).

[249] Mark A. Lemley, *The Economics of Improvement in Intellectual Property Law*, 75 TEXAS LAW REVIEW 989, 1022 (1997).

unauthorized derivative work contains some elements of the original program because its distribution would still require the original author's consent. However, it is possible to imagine a program deriving from a piece of Open Source Software but not containing any elements of the original, especially considering the existence of non-literal elements, such as the structure, sequence and organization. To the extent these elements are not protected, there is no question of infringement, copyright in new program vests in its author and he can keep it proprietary. If it was indeed derived from the copyrighted elements and as an unauthorized derivative work belonged to public domain, its use and distribution would not be encumbered. According to Nadan, even in such a case proprietarization is possible, because any person by adding even slight modifications could create the new version of a program, claim copyrights in it and decide not to reveal the upgraded source code.[250]

Therefore, the conclusion is similar to the one reached for the purpose of preemption analysis. The scope of "copyleft" clauses must be clearly determined. Only in such way there would be certainty as to when the authorization for the creation of derivative works exists and the threat of closing the Open Source program would be minimized.

4.5 Impact of Software Patents on Open Source Movement

Generally speaking, software patents are considered a serious threat for Open Source development. GNU GPL treats obtaining patent protection as yet another way of making the program proprietary and purports to make any patent "licensed for everyone's free use or not licensed at all." Stallman is aware of all the strengths of patents over copyrights but looks at it "from the point of view of its victims."[251] His arguments in *The Danger of Software Patents*, briefly summarized, are that because patents protect ideas, prohibit independent creations and

[250] Nadan, FN 200 at 371.
[251] Richard M. Stallman, *The Danger of Software Patents*, in: STALLMAN, FN 92 at 95.

require to revise the big number of already granted patents in order to avoid infringement, they make it impossible for small developers to write software. In a recent article,[252] Stallman compares patents to a minefield. Because software projects often combine thousands of ideas there is no practical possibility of making an effective search for parts that are already covered by patents; thus, "each design decision carries a risk of stepping on a patent, which can destroy [the] project."[253] According to Stallman, developers end up either with removing some software features, are forced to obtain a patent license or risk a costly procedure of overturning the blocking patent in court.

Another Open Source advocate, Bruce Perens calls upon the "incentive theory" and finds no hard evidence that software patents promote progress and in any case considers Free Software as a better means for transferring ideas to the public.[254] According to him, only the biggest market players, holding portfolios of many patents, make use of the patent system but even they use patents only defensively, to cross-license them and avoid being sued for infringement by their competitors. Thus, those developers who do not have any patents, like the Open Source Community are, in the words of Perens, "innocent bystanders injured by a war of giants"; they cannot cross-license, nor can they afford challenging a patent.

It clearly flows from the analysis of Open Source philosophy, that the development of this software does not require patent protection. Open Source developers do not intend to acquire stronger means to prevent others from using or distributing their software than those they have already had under copyright laws. As it was presented above, the legal system is used by Open Source Movement to defend the openness (free availability) of source codes

[252] Richard M. Stallman, *How to fight software patents - singly and together*, available at: http://www.newsforge.com/article.pl?sid=04/09/09/1612239.
[253] *Id.*
[254] Bruce Perens, *Software Patents v. Free Software*, available at: http://perens.com/Articles/Patents.html.

and so far the innovative licensing practices were enough to maintain it and secure the free flow of software. Relatively weak protection of copyrights allows also to design around a proprietary program, which undoubtedly is harder if it was patented. Thus, Open Source developers can possibly play only the role of patent infringers exposed to the threat of multi-million suits or forced to pay for the license. Obviously, this would strongly detract many volunteers and freelancers from contributing their code to the public.

There are; however, arguments that Open Source Software developers are actually less threatened by software patents than those who work under proprietary model because of practical impossibility of stopping the alleged infringement of the former. According to Ravicher, this is due to the fact that source codes are publicly available so instigating a lawsuit, which has to be individualized, is pointless.[255] Moreover, the "Catch-22", as he describes it, is that there is no gain in suing small individual developers, whereas attacking big and wealthy Open Source players will make them use their legal weapons against the attacker. One could add to this, that it may be easier for Open Source developers to conduct searches on already patented software before deciding what to include in their products. Because Open Source Movement has no problems in gathering volunteers, it may easily set up big teams of people searching for "patent mines", as Stallman calls them.

Open Source development model may or may not be better prepared to fight software patents. However, the true division lines between patent protection supporters and

[255] Daniel B. Ravicher, *Patents – Why Free / Open Source Software Might Have Less to Fear than Non-Free Software*, available at: http://www.groklaw.net/pdf/Patents.pdf; At the same time Ravicher himself is one of the people standing behind "patent insurance" offered by Open Source Risk Management (OSRM) and concludes that Linux has a patent risk. (Open Source Risk Management, *Results of First-Ever Linux Patent Review Announced*, available at: http://www.osriskmanagement.com/press_release_080204.pdf) OSRM has found as many as 283 patents that, if upheld as valid by the courts, could potentially be used to support patent claims against Linux. They believe; however, that no patents that have already been tested in court are infringed in the kernel.

those opposing it run between small and big developers, not necessarily between Open Source and proprietary software. Consequently, the main issue in the discussion concerns the proper means to protect the software market in general. One additional strong argument against patent protection has been raised by numerous practitioners, who believe that patent offices throughout the world lack necessary resources to diligently examine all applications. This results in granting patents for items that clearly do not fulfill requirements of novelty, non-obviousness or do not involve any inventive step.[256] But this does not result in any Open Source-specific problems, apart from subjecting it to all the drawbacks of patent offices system.

The analysis of Open Source Movement; however, provides a good point in the general discussion whether to promote software patents, such as the one taking place currently within the European Union. Perhaps, instead of discussing whether programs should be patentable "as such" or not, the drafters should ponder whether to allow for patenting information technologies at all. This is because of tremendous development of Open Source Software without the help of patents and with a remarkable technology transfer to the public at the same time. Therefore, tightening the protection and allowing private individuals for more control seems simply unnecessary.

4.6 Evaluation of Legal Controversies of Open Source Licensing

The most important conclusion of the current Chapter is that the legal situation of Open Source Software is by no means as clear and definite as the drafters of the licenses intend it to be. This is neither good nor bad, because the same applies to proprietary licensing; the law generally does not distinguish between the ideology or business model of

[256] The Foundation for a Free Information Infrastructure often informs the public about many trivial software- and Internet-related patents that are being granted throughout the world. (See http://swpat.ffii.org).

software developers. However, given the fact estimated in previous Chapter that the system designed to protect Open Source Software secures not only the private interests of developers but also the interests of the users and the society as a whole, any threat to its legal validity and enforceability should be given greater weight.

Fortunately, the practice proves the hypothesis of Moglen that licensees have more incentives to comply with Open Source licenses than to claim they are not binding;[257] thus, one should not expect too many occasions when their validity and enforceability is questioned. This is especially so because the alternatives such as first sale, fair use or implied licenses do not give users as wide a scope of rights as Open Source licenses themselves. The probability is even smaller due to the fact that the first court case directly ordering compliance with GNU GPL was recently rendered in Germany. This decision should be evaluated as a big step towards the increase of Open Source legal certainty, although it remains remarkable that most of such disputes are resolved by negotiations.

Still, the analysis presented above proves that there are possibilities for putting Open Source Movement under the legal threat. The major such possibility is the invalidation of the whole license on the basis of lack of clear communications between the contracting parties. This again is not a specific problem of Open Source licensing; however, there is no preferential treatment for it under law. Therefore, the developers should take care to expressly communicate terms of licenses to make them binding under every legal regime discussed in this Chapter.

Apart from various contract formation rules, the institutes relating to copyright law were also analyzed above. The conclusion from that part is that the licenses should not be

[257] See FN 156 and accompanying text.

preempted or considered to constitute a copyright misuse if clauses such as “copyleft” are reasonably construed. Moreover, the recognition of author's moral rights could very well supplement the goals aimed at by Open Source licensing. Most importantly, the legal analysis is backed up with policy considerations, because the system for the protection of Open Source Software is in line with the underlying idea of intellectual property laws – to balance private and public interests and may even be considered to align these interests.

Therefore, there seem to be no serious threats to any of the discussed Open Source licenses as a whole; however, there may be legal ways to avoid their particular clauses. One of the very important examples discussed in current Chapter is “copyleft”. Although parties might agree that it covers derivative works only this does not answer the question what exactly its legal effect is. It seems that courts may have many problems in determining legal relationships between programs, the development techniques of which evolve constantly, whereas the law and vague licenses do not give necessary guidelines. Apart from technical difficulties, the legal situation of program derivatives varies throughout jurisdictions, which supports the hypothesis of Nadan that there are possibilities of evading “copyleft” clauses.[258]

Another important legal controversy surrounding software licensing is the validity and enforceability of various liability limitations and warranty disclaimers. This issue is not specific to Open Source Movement; still, it may have bigger impact on its participants than on proprietary developers because having to pay damages for defective software is undoubtedly more burdensome for those who contribute their code for free. However, if the licensors forget to expressly point out limitations and disclaimers to the users or draft them carelessly, they may fall under liability regimes provided for in such laws as UCC, CISG or

[258] See FN 250 and accompanying text.

various European consumer protection directives. Here, the black letter law may prevail over otherwise brilliant economic and axiological arguments for upholding the licensor's immunity.

The last issue discussed in this Chapter was the impact of software patents on Open Source Movement. The analysis supports the argument that although there are considerably more opponents to granting patents for software in this movement than anywhere else, it is rather a case between the large and the small, regardless of their ideology or business model. However, the conclusion for the purpose of this paper is that Open Source Movement does not need patent protection, because it managed to develop and maintain remarkable software projects without the use of it. More importantly, all this has been done by extending access rights of third parties, preserving public interest and contradicting the hypothesis that more private control is required for successful software development. Therefore, if Open Source proves to be capable of sustaining its success, software patents will be considered unnecessary.

The major purpose of this Chapter was to evaluate the validity and enforceability of Open Source licenses and to answer the question whether they allow Open Source Movement to realize its goals. Summing up, although there is legal uncertainty connected with this way of licensing, the risk cannot be considered as substantially higher than the one surrounding proprietary licenses; thus, all developers can only benefit from increasing their legal diligence maybe even over the level of care they put into software design. The final conclusion as to Open Source-specific legal controversies should be that the use of law by the movement is in line with the doctrine of intellectual property laws and can even trigger the reconsideration of some of the traditional premises as to what the black letter law should be.

Conclusion

The purpose of this paper as stated in the Introduction, was to observe, analyze and comment on the reciprocal relation between Open Source Movement and law. This was done by contributing towards two major debates currently taking place. Firstly, the one about the role of law in the movement, in which also the questions about the theory and ideology of intellectual property laws are considered. Secondly, the more detailed one, devoted to the legal effect of Open Source licenses or their particular clauses faced with the laws of specific jurisdictions.

The analysis started with a brief overview of the general rules for the protection and use of computer programs, the presentation of the default legal system, its theoretical base and practical application in the proprietary model. That constituted the necessary background for the discussion on the very subject matter of the paper. Together with legal issues, the economic, technical and social aspects of software market in general and of Open Source Software development were analyzed. Therefore, the legal debate was put in the important context of the rationales, ideology and interests of the parties involved. Nevertheless, the paper focused on the legal issues in both theoretical and practical aspects.

All the conclusions and points made throughout the paper can be summarized as follows. One of the first findings of this paper was that none of the regimes of legal protection fits computer programs ideally. This conclusion was reached after confronting the general framework of the legal protection of computer programs with their specific technical and economic features. Although it should not be understood that all of the regimes described are inherently bad for computer programs, it logically explains why software developers undertake efforts to tailor the legal protection better through the use of private contracts –

licensing. The developers of proprietary software draft their licenses very restrictively and additionally undertake efforts to secure for themselves the benefits of even stricter regime – patents.

It was concluded that Open Source Movement should be confronted with proprietary software, as it uses the law to the opposite of proprietary licensing. At the same time it is clearly not correct to equate Open Source and public domain software. This is also because of the active use it makes of intellectual property laws. But not only legal factors influence Open Source Software. The movement has a rich and interesting history and it encompasses a wide variety of software products. More importantly, it evolved from the cooperation between individuals and social groups having diverse interests and motivations. The analysis of all these factors led to the conclusion that the system designed by them is specific and unique. As a whole, it remains remarkably flexible and allows the moral crusaders to spread their idea, the pragmatists to profit from their marketing model and at the same time it protects the users' from being deprived of their access rights and turned into passive consumers.

Although ethical norms and custom that evolved in the community form an important element of the system, both moral and pragmatic goals of the participants in Open Source Movement could not be realized without the use of law. This conclusion was reached mainly after analyzing the content of model Open Source licenses, which grant the licensees a broad scope of rights. Law is gaining even more importance in Open Source Movement recently, as the developers organize themselves in more formal structures and provide for quite sophisticated intellectual property rights management schemes. The analysis of this specific system designed to protect and allow for the use of computer programs led to yet one more conclusion that it does not conflict with the current theoretical and axiological framework of intellectual property law, which tries to make up for the failures of public goods markets by striking the right balance between private and public interests. Moreover, the findings of this paper support the statement that not only the system for the protection of Open Source Software withstands the scrutiny of intellectual property legal theory, but also it may cause the reconsideration of arguments of proprietary developers calling for stronger protection.

Additionally to the conclusion that Open Source Movement deserves protection, the detailed analysis of relevant legal rules and doctrines of the U.S. and EU proved that current legal systems discussed in this paper do indeed protect it to a great extent. Although there are some controversies around the validity and enforceability of the licenses or their particular clauses, many of them remain the basis for theoretical discussion only as there are simply more incentives to comply than to infringe. Moreover, most of the legal threats pointed out in this paper do not concern Open Source in particular, but have to be considered by proprietary vendors as well. However, as it was already raised, the social value of Open Source Movement requires to attribute a greater weight to all these threats and obliges the drafters of licenses to a higher diligence. Therefore, the licensors should pay particular attention to contract formation issues and secure for valid communications between the parties. Care should also be taken while drafting “copyleft” clauses to clarify their scope and secure them against diversities of national laws. Moreover, Open Source developers should take under account the existence of default liability and warranty regimes under various jurisdictions if they intend to maintain the cosmopolitan nature of their projects.

Given the breadth of the subject, its complex nature as well as social importance, all of the conclusions hereto should be treated as the invitation to further research. It may focus on the specific regulations of any given national jurisdiction, such as particular copyright or contract law provisions and doctrines. For example, the important question not dealt with in this paper is whether model Open Source licenses should be adjusted by translating them into national legal languages. On the general level, the analysis of the relationship between Open Source computer programs and law may definitely be continued because the system designed for their legal protection and use is still evolving. Perhaps, the continuation of the movement's success may trigger ideas to redraft intellectual property laws but even in the current legal situation there is still much left to discuss.

Bibliography

Authoritative Writings

Daniel J. M. Attridge, *Challenging Claims! Patenting Computer Programs in Europe and the USA*, 1 Intellectual Property Quarterly 22 (2001)

David Bainbridge, Software Copyright Law (Butterworths, London, Edinburgh, Dublin, 4th ed., 1999)

Lorin Brennan, *Why Article 2 Cannot Apply to Software Transactions*, 38 Duquesne Law Review 459 (2000)

M. Byrska, *Prawne aspekty modyfikowania programu komputerowego [Legal aspects of modifying computer program]*, 4 Kwartalnik Prawa Prywatnego [Private Law Quarterly] 693 (1996)

Trevor Cox, *Chaos versus uniformity: the divergent views of software in the International Community*, 4 Vindobona Journal of International Commercial Law and Arbitration 3 (2000)

Victoria A. Cundiff, *Protecting Computer Software as a Trade Secret*, in: 507 Practising Law Institute, 18th Annual Institute on Computer Law 761 (February 1998)

Estelle Derclaye, *Software Copyright Protection: Can Europe Learn from American Case Law?* Part 1, 1 European Intellectual Property Review 10 (2000); Part 2, 2 European Intellectual Property Review 56 (2000)

Chris DiBona, Sam Ockman, and Mark Stone, eds., Open Sources: Voices from the Open Source Revolution (O'Reilly, 1999)

Frank Diedrich, *The CISG and Computer Software Revisited*, 6 Vindibona Journal Supplement 55 (2002)

Joseph Drexl, What is protected in a computer program?: Copyright protection in the United States and Europe (VCH Verlagsgesellschaft mbH, Weinheim, New York, 1994)

Garry L. Founds, *Shrinkwrap and Clickwrap Agreements: 2B or not 2B?*, 52 Federal Communications Law Journal 99 (1999)

Paul Goldstein, Copyright (Aspen Law & Business, New York, 2nd ed. 1998)

Robert W. Gomulkiewicz, *De-bugging Open Source Software Licensing*, 64 University of Pittsburgh Law Review 75 (2002)

Kevin W. Grierson, *Enforceability of "Clickwrap" or "Shrinkwrap" Agreements Common in Computer Software, Hardware, and Internet Transactions*, 106 American Law Reports 5th 309

Alois Valerian Gross, *What is Computer "Trade Secret" under State Law*, 53 American Law Reports 4th 1046

Robert Hillman, *Rolling Contracts*, 71 Fordham Law Review 743 (2002)

Sean Hogle, *Unauthorized Derivative Source Code*, 18 No. 5 Computer and Internet Lawyer 1 (2001)

Dennis S. Karjala, *Federal Preemption of Shrinkwrap and On-Line Licenses*, 22 University of Dayton Law Review 511 (1997)

D. S. Karjala, *Recent United States and International Development in Software Protection*, Part 1, 1 European Intellectual Property Review 13 (1994), Part 2, 2 European Intellectual Property Review 58 (1994)

Stephen T. Keohane, *Mass Market Licensing*, in: 652 Practising Law Institute, Patent & Technology Licensing 2002 269 (2002)

Irene Kosturakis, *Software Licensing and UCITA*, in: 762 Practising Law Institute, Understanding the Intellectual Property License 2003 437 (2004)

Mark A. Lemley, *Beyond Preemption: The Law and Policy of Intellectual Property Licensing*, 87 California Law Review 111 (1999)

Mark A. Lemley, *The Economics of Improvement in Intellectual Property Law*, 75 Texas Law Review 989 (1997)

Lawrence Lessig, The Future of Ideas: the fate of the commons in a connected world (Random House, New York, 2001)

Ian J. Lloyd, Information Technology Law (Butterworths, London, Edinburgh, Dublin, 3rd ed., 2000)

Axel Metzger, Till Jaeger, *Open Source Software and German Copyright Law*, 32 IIC International Review of Industrial Property and Copyright Law 52 (2001)

Arthur R. Miller, *Copyright protection for computer programs, databases, and computer-generated works: is anything new since CONTU?*, 106 Harvard Law Review 977 (1993)

Erik Mosesson ed., Software Procurement Nordic Yearbook of Law and Informatics 1992 (Norstedts Juridik, Stockholm, 1992)

Christian H. Nadan, *Open Source Licensing: Virus or Virtue?* 10 Texas Intellectual Property Law Journal 349 (2002)

Kenneth Nichols, Inventing software: the rise of “computer-related” patents (Quorum Books, Westport, 1998)

Raymond T. Nimmer, The Law of Computer Technology (West Group, rev. ed. 1999)

Raymond T. Nimmer, *Breaking Barriers: The Relations Between Contract and Intellectual Property Law*, 13 Berkeley Technology Law Journal 827 (1998)

A. Nowicka, Prawnoautorska i patentowa ochrona programów komputerowych [Copyright and patent protection of computer programs] (Dom Wydawniczy ABC, Warszawa, 1995)

A. Samuel Oddi, *An Uneasier Case for Copyright than for Patent Protection of Computer Programs*, 72 Nebraska Law Review 351 (1993)

Christopher L. Ogden, *Patentability of Algorithms After State Street Bank: The Death of the Physicality Requirement*, No. 10 Vol. 82 Journal of Patent and Trademark Office Society 721 (2000)

Shawn W. Potter, *Opening Up to Open Source*, 6 Richmond Journal of Law and Technology 24 (2000)

Carey R. Ramos, David S. Berlin, *Three Ways to Protect Computer Coftware*, 16 No. 1 COMPUTER LAWYER 16 (1999)

DIANE ROWLAND, ELIZABETH MACDONALD, INFORMATION TECHNOLOGY LAW (Cavendish Publishing Ltd, London, Sydney, 2nd ed., 1997)

Pamela Samuelson *et. al.*, *A Manifesto Concerning The Legal Protection Of Computer Programs*, COLUMBIA LAW REVIEW, December 1994, at 2308

Daniele Schiuma, *TRIPS and Exclusion of Software "as Such" from Patentability*, No.1 Vol. 31 IIC INTERNATIONAL REVIEW OF INDUSTRIAL PROPERTY AND COPYRIGHT LAW 36 (2000)

Yannis Skulikaris, *Software-Related Inventions and Business-Related Inventions; A review of practice and case law in U.S. and Europe*, PATENT WORLD, February 2001, at 26.

RICHARD M. STALLMAN, FREE SOFTWARE, FREE SOCIETY: SELECTED ESSAYS OF RICHARD M. STALLMAN (GNU Press, Boston, 2002)

Peter Toren, *Software and Business Methods are Patentable in the U.S. (Get over it)*, PATENT WORLD, September 2000, at 7

R C Tripathi *et al.*, *Patenting of Computer Software: Status and Approach*, Vol. 7 JOURNAL OF INTELLECTUAL PROPERTY RIGHTS 128 (2002)

Julian Velasco, *The copyrightability of non-literal elements of computer programs*, COLUMBIA LAW REVIEW, January 1994, at 242

F. Warren-Boulton *et. al.*, *Economics of intellectual property protection for software: The proper role for copyright*, 3 no. 2 STANDARD VIEW 68 (June 1995)

ROBERT YOUNG & WENDY GOLDMAN ROHM, UNDER THE RADAR: HOW RED HAT CHANGED THE SOFTWARE BUSINESS – AND TOOK MICROSOFT BY SURPRISE (Coriolis Group Books, Scottsdale, Arizona 1999)

Authoritative Writings – Internet Sources

Agency for the development of electronic administration (ADAE), *Guide to Choosing and Using Free Software Licenses for Government and Public Sector Entities*, available at: http://www.adae.gouv.fr/upload/documents/free_software_guide.pdf

The Apache Software Foundation, *About the Apache HTTP Server Project*, available at: http://httpd.apache.org/ABOUT_APACHE.html

The Apache Software Foundation, *Home Page*, available at: http://www.apache.org/

The Apache Software Foundation, *How the ASF works*, available at: http://www.apache.org/foundation/how-it-works.html

Piotr Bolek, *Forum Rozwoju Wolnego Oprogramowania [Forum for the Free Software Development]*, available at: http://frwo.linux.org.pl/ (in Polish)

CISG-AC, *Opinion no. 1, Electronic Communications under CISG*, 15 August 2003, available at: http://www.cisg.law.pace.edu/cisg/CISG-AC-op1.html

Debian Documentation Team, *A Brief History of Debian*, available at: http://www.debian.org/doc/manuals/project-history/

European Communities, *Europa – Information Society – Free & Open Source Software – Introduction*, available at: http://europa.eu.int/information_society/activities/opensource/index_en.htm

Foundation for Free Information Infrastructure, *Software Patents in Europe: A Short Overview*, available at: http://swpat.ffii.org/lisri/intro/index.en.html

Free Software Foundation, *Frequently Asked Questions about the GNU GPL*, http://www.fsf.org/licenses/gpl-faq.html

Free Software Foundation, *Various Licenses and Comments about Them*, available at: http://www.fsf.org/licenses/license-list.html

Eben Moglen, *Free Software Matters: Enforcing the GPL, I*, available at: http://moglen.law.columbia.edu/publications/lu-12.html

Eben Moglen, *Free Software Matters: Enforcing the GPL, II*, available at: http://emoglen.law.columbia.edu/publications/lu-13.html

The Mozilla Organization, *Mozilla Roles and Responsibilities*, available at: http://www.mozilla.org/about/roles.html

The Mozilla Organization, *mozilla.org License Policy*, available at: http://www.mozilla.org/MPL/license-policy.html

Netscape, *Netscape Announces Plans to Make Next-Generation Communicator Source Code*, available at: http://wp.netscape.com/newsref/pr/newsrelease558.html

Netscape, *Netscape Launches Aggressive 'Unlimited Distribution' Program For New Free Client Software*, available at: http://wp.netscape.com/newsref/pr/newsrelease560.html

Open Source Initiative, *The Open Source Definition,* available at: http://www.opensource.org/docs/definition.php

Open Source Initiative, *Frequently Asked Questions*, available at: http://www.opensource.org/advocacy/faq.php

Open Source Initiative, *History of the OSI*, available at: http://www.opensource.org/docs/history.php

Open Source Initiative, *Open Source Case for Business*, available at: http://www.opensource.org/advocacy/case_for_business.php

Open Source Initiative, *The Open Source Case for Customers*, available at: http://www.opensource.org/advocacy/case_for_customers.php

Open Source Initiative, *The Open Source Case for Hackers*, available at: http://www.opensource.org/advocacy/case_for_hackers.php

Open Source Risk Management, *Results of First-Ever Linux Patent Review Announced*, available at: http://www.osriskmanagement.com/press_release_080204.pdf

Open Source Technology Group, *SourceForge.net: Software Map*, available at: http://sourceforge.net/softwaremap/trove_list.php?form_cat=14

Bruce Perens, *Software Patents v. Free Software*, available at: http://perens.com/Articles/Patents.html

Daniel B. Ravicher, *Patents – Why Free / Open Source Software Might Have Less to Fear than Non-Free Software*, available at: http://www.groklaw.net/pdf/Patents.pdf

Eric S. Raymond, *Homesteading the Noosphere*, available at: http://www.catb.org/~esr/writings/cathedral-bazaar/homesteading/

Eric S. Raymond, *The Magic Cauldron*, available at: http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/

Larry Rosen, *The Unreasonable Fear of Infection*, available at: http://rosenlaw.com/html/GPL.PDF

Software in the Public Interest, *Debian Social Contract*, available at: http://www.debian.org/social_contract.en.html

Richard M. Stallman, *How to fight software patents - singly and together*, available at: http://www.newsforge.com/article.pl?sid=04/09/09/1612239

Sun Microsystems, *Contributing to OpenOffice.org*, available at: http://www.openoffice.org/contributing.html

Sun Microsystems, *download: Download Central*, http://download.openoffice.org/1.1.2/index.html

Sun Microsystems, *FAQs*, available at: http://www.openoffice.org/FAQs/faq-licensing.html

Sun Microsystems, *License Page*, available at: http://www.openoffice.org/license.html

Sun Microsystems, *OpenOffice.org 1.1 Product Description*, available at: http://www.openoffice.org/product/

Linus Torvalds, *Note to Linux Kernel*, available at: http://www.linux.de/linux/gnu.html

Harald Welte, *netfilter project GPL settlement with Securepoint*, available at: http://www.netfilter.org/news/2004-03-25-securepoint-gpl.html

werk21, *Bundestux – Pinguine ins Amt! [Bundestux – Penguins to the Office!]*, available at: http://www.bundestux.de/

International Legal Documents

Berne Convention for the Protection of Literary and Artistic Works (1886, Paris Act of 1971)

Convention on the Law Applicable to Contractual Obligations (Rome Convention, 1980)

European Patent Convention (1973)

Guidelines for Examination at the European Patent Office (2003)

UN Convention on the International Sale of Goods (CISG, Vienna Convention, 1980)

WIPO Copyright Treaty (1996)

WIPO Model Provisions on the Protection of Computer Software (1978)

WTO Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS) (1994)

American Legal Documents

Constitution of the United States of America

Copyright Act of the United States of America, 17 U.S.C. Sec. 101-1332 (1976, as amended)

Uniform Commercial Code

Uniform Computer Information Transactions Act (Approved Official Draft), available at: http://www.law.upenn.edu/bll/ulc/ucita/2002final.htm

Legal Documents of European Union

Directive on certain aspects of the sale of consumer goods and associated guarantees, 1999/44/EC, OJ L 171, 07/7/1999 P. 12

Directive on certain legal aspects of information society services, in particular electronic commerce, in the Internal Market (E-Commerce Directive), 2000/31/EC, OJ L 178, 17/07/2000, P. 1

Directive on the approximation of the laws, regulations and administrative provisions of the Member States concerning liability for defective products, 85/374/EEC, OJ L 210, 07/08/1985 P. 29

Directive on the legal protection of computer programs, 91/250/EEC, OJ L122, 17/05/1991 P. 42

Directive on the protection of consumers in respect of distance contracts, 97/7/EC, OJ L 144, 04/06/1997, P. 19.

Directive on unfair terms in consumer contracts, 93/13/EEC, OJ L 095, 21/04/1993 P. 29

Proposal for the Directive on the legal protection of computer programs, COM (88) 816 final – SYN 183 [1989] OJ C91/9

Proposal for the Directive on the patentability of computer-implemented inventions, 6580/02 PI 10 CODEC 242

Regulation on the application of Article 81(3) of the Treaty to categories of technology transfer agreements (Technology Transfer Regulation), 2004/772/EC, OJ L 123, 27/04/2004, P. 11

Other Legal Documents

Apache License Version 2.0, available at: http://www.apache.org/licenses/LICENSE-2.0

BSD License, available at: http://www.xfree86.org/3.3.6/COPYRIGHT2.html

GNU General Public License, Version 2.0, available at: http://www.fsf.org/licenses/gpl.txt

GNU Lesser General Public License, Version 2.1, available at: http://www.fsf.org/copyleft/lesser.txt

Individual Contributor License Agreement (Apache), available at: http://www.apache.org/licenses/cla.txt

Mozilla Public License (Annotated Version 1.1), available at: http://www.mozilla.org/MPL/MPL-1.1-annotated.html

OpenOffice.org Open Source Project Joint Copyright Assignment by Contributor, available at: http://www.openoffice.org/licenses/jca.pdf

Sun Industry Standards Source License – Version 1.1, available at: http://www.openoffice.org/licenses/sissl_license.html.

Table of Cases

American Cases

Adobe Systems, Inc. v. One Stop Micro, Inc. 84 F. Supp.2d 1086 (N.D. Cal. 2000).

Apple Computer, Inc. v. Franklin Computer Corp. 714 F.2d 1240 (3d Cir.1983), cert. dismissed 464 U.S. 1033 (1984).

Architectonics, Inc. v. Control Systems, Inc. 935 F. Supp. 425 (S.D. N.Y. 1996).

Berthold Types Ltd. v. Adobe Systems, Inc. 101 F. Supp.2d 697 (E.D. Ill 2000).

Brown Bag Software, Inc. v. Symanthec Corp. 960 F 2d 1465 (9th Cir 1992).

Caspi v. Microsoft Network L.L.C. 732 A.2d 528 (N.J. Super Ct. App. Div. 1999), cert. denied, 743 A.2d. 851 (1999).

CMS Software Design Sys., Inc. v. Info Designs, Inc. 785 F.2d 1246 (5th Cir.1986).

Computer Associates, Inc. v. Altai, Inc. 982 F.2d 693 (2nd Cir 1992).

EF Cultural Travel BV v. Zefer Corp. 2003 U.S. App. LEXIS 1336 (1st Cir. 2003).

Hines v. Davidowitz, 312 U.S. 52, 67 (1941).

Hotmail Corp. v. Van$ Money Pie, Inc. 47 U.S.P.Q.2d 1020 (N.D. Cal 1998).

I.LAN Systems, Inc. v. Netscout Service Level Corp. 183 F.Supp.2d 328 (D. Mass. 2002).

Lasercomb America, Inc. v. Reynolds 911 F.2d 970 (4th Cir. 1990).

Lotus Development Corp. v. Paperback Software Int. 740 F. Supp. 37 (D. Mass. 1990).

M.A. Mortenson Co. v. Timberline Software Corp. 970 P.2d 803 (Wash. Ct. App. 1999), aff'd, 998 P.2d 305 (Wash. 2000).

Microstar v. Formgen, Inc. 942 F. Supp. 1312 (S.D. Cal. 1996), aff'd in part, rev'd in part, 154 F.3d 1107, 48 U.S.P.Q.2d (BNA) 1026 (9th Cir. 1998).

Novell, Inc. v. Network Trade Center, Inc. 25 F. Supp. 2d 1218 (D. Utah 1997).

Planetary Motion, Inc. v. Techsplosion, Inc. 261 F.3d 1188 (11th Circ. (Fla.), 2001).

ProCD, Inc. v. Zeidenberg 86 F.3d 1447 (7th Cir. 1996).

Progress Software Corp. v. MySQL AB 195 F. Supp. 2d 328 (D. Mass. 2002).

Register.com, Inc. v. Verio, Inc. 2004 WL 103400, 69 U.S.P.Q.2d 1545 (C.A.2 (N.Y.), 2004).

SCO Group, Inc., The v. International Business Machines, No. 2:03cv0294 (Plaintiff's Amended Complaint) (D. Utah, filed June 16, 2003).

Softman Products Co., LLC v. Adobe Systems, Inc. 171 F. Supp. 2d 1075 (C.D. Cal. 2001).

Sony Corp. of Am. v. Universal City Studios, Inc. 464 U.S. 417 (1984).

Specht v. Netscape Communications Corp. 306 F.3d 17 (2nd Cir. (N.Y.) 2002).

State Street Bank & Trust v. Signature Financial Services, 149 F.3d 1368 (Fed. Cir. 1998), cert. denied 119 S.Ct. 851, (1999).

Step-Saver Data Sys., Inc. v. Wyse Tech. 939 F.2d 91 (3d Cir. 1991).

Storm Impact, Inc. v. Software of the Month Club, 13 F.Supp. 2d 782 (N.D. Ill. 1998).

Ticketmaster Corp. v. Tickets.com, Inc. 54 U.S.P.Q.2d (BNA) 1344 (C.D. Cal. 2000).

Unix System Laboratories v. Berkeley Software Design, Inc. 1993 Copr.L.Dec. P 27,075, (27 U.S.P.Q.2d 1721); 1993 Copr.L.Dec. P 27,166, (86 Ed. Law Rep. 738, 29 U.S.P.Q.2d 1561).

Whelan Associates, Inc. v. Jaslow Dental Laboratory Inc. 797 F.2d 1222 (3d Cir. 1987).

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

German Case

Landgericht München, 19.5.2004, 21 O 6123/04.

European Patent Office

International Business Machines, Corp./Computer program product, Decision of Technical Board of Appeal 3.5.1 dated 1 July 1998, T 1173/97 (OJ 10/1999, 609)