FACULTY OF LAW
LUND UNIVERSITY

AVE-LIIS SALUVEER

THE CONCEPT OF DERIVATIVE WORKS UNDER THE EUROPEAN COPYRIGHT LAW IN RELATION TO THE DIGITAL ERA: FREE AND OPEN SOURCE SOFTWARE LICENSING

JAEM03 MASTER THESIS

EUROPEAN BUSINESS LAW
30 higher education credits

Supervisor
BJÖRN LUNDQVIST

Term
SPRING 2014

SUMMARY

As today's society is majorly influenced by the technological inputs and its developments, it is necessary to have a clear understanding of the legal concepts in relation to this field. This thesis focuses on software programs. Namely, this is a paper assessing the matter of derivative works in relation to computer programs. The concept of derivative work itself involves a level of uncertainty, but the author of this thesis aims to propose possible methods in order to distinguish a derived work from an original within the field of computer programs.

Although the motion of free software promotion is not new, the field of free and open source software has particularly grown over the past decades. The definition of derivative works plays a significant legal role also in relation to this matter. Therefore, this thesis covers the matter of free and open source software licensing, while mainly focusing on the sections relating to the adaptations of the original work. Conclusive legal analyses are given through an evaluation of the most common FOSS licences and their core values in regards to derivative works.

PREFACE

> "Only one thing is impossible for God: to find sense in any copyright law on the planet."
>
> - Mark Twain

Yet, as many Master of law students and their final works, trying to find sense has as well been my aim of this thesis. I have gathered my research on the matter of derivative works in copyright law in order to hopefully expand some clarity in this aspect's relation to computer programs.

This thesis is one of the fruits of my studies in the Lund University over the past two years and was inspired by an internship position in an IT law sector.

I would like to express my gratitude to the university, my supervisor as well as the lecturers, who have openly shared their wisdom and experience. I am also extremely thankful for the close people around me. They have withstood this busy and nervous period in my life, while giving me enormous support and occasional hugs when needed.

I hope you will enjoy reading this thesis.

Ave-Liis Saluveer

Lund, June 2014

ABBREVIATIONS

AFC	Abstraction-Filtration-Comparison
BSD	Berkeley Software Distribution
CJEU	The Court of Justice of the European Union
DRM	Digital Restriction Management
EU	European Union
EUPL	European Union Public Licence
FIPR	Foundation of Information Policy Research
FOSS	Free and Open Source Software
FSF	Free Software Foundation
GPL	General Public Licence
GPL v3	version 3 of the General Public Licence
LGPL	Lesser General Public Licence
PC	Personal Computer
POSAR	Planning-Operationalization-Separation-Analysis-Reporting
WIPO	World Intellectual Property Organization
WPL	World Programming Ltd.

ABBREVATIONS FOR LEGISLATIVE DOCUMENTS

Berne Convention	Berne Convention for the Protection of Literary and Artistic Works
InfoSoc Directive	Directive 2001/29/EC of the European Parliament and of the Council of 22 May 2001 on the harmonisation of certain aspects of copyright and related rights in the information society
Software Directive	Directive 2009/24/EC of the European Parliament and of the Council of 23 April 2009 on the protection of computer programs
TRIPS Agreement	Agreement on Trade-Related Aspects of Intellectual Property Rights

Table of Contents

1. Introduction

In today's society, largely influenced by digital and technological inputs, free and open source software (hereinafter "FOSS") has become an interesting phenomenon that has gained increasing popularity among all continents. Although the notion of free software is not necessarily new, the field has grown drastically over the past decades. While the concept has reached a critical mass over the past year, there are still some legal uncertainties attached to the field of software licensing.

One of the ambiguous matters lies in possible derivations of FOSS and this thesis will aim to give a better understanding of the topic. By its very concept, FOSS users are encouraged to modify and improve the source code and redistribute it back to the community for further development for all of their users. Therefore, the legal perspective of derivative works is of major importance within the technological community. It is thus vital to clarify the distinction between the original work, potential derivative works as well as software that constitutes as a new copyrighted work.

1.1. Purpose

The thesis will be focused on giving an in-depth analysis on derivative works in the copyright law, while putting focus on the legal concept of derived works in relation to software programs. This thesis also emphasises the problematic uncertainty in regards to derivative works when derived from FOSS. The legal analysis is given through an evaluation of the most common FOSS licences and their core values in regards to derivative works.

1.2. Method and Material

The thesis is written taken into consideration the traditional dogmatic approach in legal research, while exercising comparative analysis in relation to different FOSS licences. The author has sought to strike a balance between presenting merely academic argumentations and illustrating the matter with practical elements through FOSS licences.

Sources of law derive from legal instruments, such as the international framework on copyright law and legislation on the European level. Due to the global spread of software

projects and the legal aspects that follow in conjunction with the vast development of technological society, the assessment has been done on the most common licences in relation to this matter. Additionally, other sources of information include relevant case law, legal journals and articles.

1.3. Delimitations

In order to provide a satisfactory conclusion to the specified research, there are several limitations that need to be recognised. This thesis is limited to defining derivative works in relation to computer programs, while the matter does not cover the legal issues in relation to decompilation. Furthermore, in order to assess both the theoretical as well as practical aspect in relation to the topic, the author has provided an analysis in relation to different FOSS licences. However, the purpose of this thesis is not to give a full statement of all aspects in FOSS licences and is therefore limited to assessing three main licences relevant to the topic at hand: General Public Licence, BSD Licence and the European Union Public Licence.

1.4. Research Questions

When assessing derivative works in relation to FOSS, it is important to firstly stress that if a work constitutes a derivative from the original product, that work generally needs to be licensed under the relevant copyleft-licence. The underlying question of this thesis is to define what works under the given circumstances need to be licensed under such copyleft-licence and how the differentiation of derived and original works could be made. To fully comprehend the issue at hand, it becomes necessary to examine the backgrounds, legal frameworks and definitions of such terms as “derivative work” and “computer program” in the legal context, while also emphasizing the free and open source software movement with the common licence agreements. I will seek to clarify what constitutes a derivative work, while proposing how the differentiation between an original work and derived work could be assessed in the case of software programs. The assessment will be followed by an analysis of different FOSS licence agreements focusing on the notions of derivative works, while also bringing out the specific peculiarities in different FOSS licences. The following questions will be researched and answered in this thesis:

- What are derivative works in the copyright law?
- How does the law protect derivative works?

- Is the programming architecture the only criteria when determining what a derivative work is?
- What is the significance of the legal definition of computer programs?
- How should derivative works be defined in matters of computer programs?
- What are free and open source software programs and what is their relation to derivative works?
- How are free and open source software programs legally protected?
- What requirements are set on derivative works under the most common FOSS licence agreements?

1.5. Disposition

The thesis has been divided into eight parts. Firstly, the thesis gives the general background on the definition of derivative works and the origin of this concept, while assessment has been done in relation to the international framework as well as in relation to the European legislation. Next a synopsis of the central topical issue of computer programs and the legal understanding behind this term is set out. Thirdly, the development of free and open source software has been stated in order to get a full understanding of the issue at hand. After the information feature follows a legal assessment of the derivative works in relation to software programs, where different possible approaches on distinguishing derivative works from original works, in the field of computer programs, are given. Next, an analysis of the peculiarities of different FOSS licences are given, while focus is put on matters of derivative works and the "copyleft" principle. The thesis ends with a conclusive assessment aiming to clarify the problematic issue on how derivative works could be defined in computer programs, while being finalised with a brief summary and the author's analysis.

2. Derivative Works in Copyright Law

2.1. The Definition of Derivative Works

The term of "derivative works" is initially coming from the United States' legal concept and is defined as the following:

> "derivative work" is a work based upon one or more pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, consideration,

> 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".[1]

It is clear from the wording of the definition that the work has to be based upon one or more pre-existing copyrighted works in order to fall within the scope of derivation. The U.S. case-law has further elaborated on the definition stating that in matter of computer programs, the program has to be substantially similar to the copyrighted work while including a portion of that work in its final product.[2] In *Unites States v. Manzer*, it was held that 70% similarity in the code is to be considered sufficient to claim the work to be derivative.[3]

As to the aspect in Europe, the concept of derivative works is not straightforward either. For the comparison, the exact corresponding definition as the U.S. legal concept is not commonly used. In Europe, the term can be related to the adaptation matters in the relevant legislations. The Foundation of Information Policy Research (hereinafter "FIPR") however did define derivative works in its guidance to implementing the EU Copyright Directive. Derivative works under that were to be considered works that were based upon the original work or upon the original work and other pre-existing works, such as translation, musical arrangement, dramatization, sound recording or any other form in which the original work may be recast, transformed or adapted.[4] This definition is essentially a copy of the U.S definition and emphasises further the matter that the derived work has to be based upon the original work in order to fall within the scope of derivative works. In principle, both EU and U.S. recognise derivative work rights as part of copyright that does also include the right to alter content of an initial work, take extracts from the original work, combine them, translate them, or otherwise create a new work from the existing work, an owner of the creative work does have the absolute right to create such derivative works.[5]

---

[1] U.S Code Title 17 Chapter 1 § 101.
[2] Case Litchfield v. *Spielberg*, 736 F. 2d 1352, 9th Cir. (1984).
[3] Case *United States v. Manzer*, 69 F.3d 222, 8th Cir., (1995).
[4] I. Brown, *Implementing the EU Copyright Directive*, (2003). Available at: http://www.fipr.org/copyright/guide/eucd-guide.pdf (Last accessed: 26.05.2014).
[5] F. Busnello, O. Romanenko, *Copyright vs. Fan-made Derivative Works: Unresolvable Conflict or Unavoidable Impulse for Reform of the Present Copyright System*, (2012). Available at: http://www.academia.edu/1255504/Copyright_vs._Fan-made_Derivative_Works_unresolvable_conflict_or_unavoidable_impulse_for_reform_of_the_present_copyright_system (Last accessed: 26.05.2014).

In essence, an otherwise copyrightable work has to satisfy two requirements in order to be considered as a derivative work.[6] Firstly, the work has to borrow from another work and secondly, the work has to recast, transform or adapt the work upon which the new work is based.[7] Although the standard definition is to an extent understandable, the uncertainty lies in the matter that there is no specification as to how different a derivative work has to be from the original work to merit copyright protection. If one is to interpret an original work that has fallen for an example into the public domain to an extent, the level of originality in order for the new work to fall within the scope of copyright protection is unclear and principally rather complex to be fixed, since the area itself is broad and the variety of possibilities for modifications are immense.

From the abovementioned, it is clear that "derivative work" stands for a concept of work that has a relation to a previously copyrighted work, although the exact level of originality when forming a derivative piece involves a matter of uncertainty. In principle, there is an obligation for the derivative work creator to obtain prior permission from the proprietor of the initial work when creating such derived works, except when the basis for the derived piece belongs to the public domain. The song "Love Me Tender" is credited as being the song written by Elvis Presley and Vera Matson, although it was derived from a public domain song named "Aura Lee".[8] No permission was required to publish the derivative piece and the authors of "Love Me Tender" benefitted from the profits. However, Puff Daddy's song "I'll Be Missing You" is a musical derivation based on Sting's "Every Breath You Take" and thus the latter consequently had the right to either give permission to use the original or not, while owning part of the derivative work and subsequently making profit from the derivative work as well as from the original piece.[9] Therefore, there can be explicit occasions where the creation of the derivative work is not questioned, but just as in the field of music, in the technological matters the majority of possible derived works are not as clear-cut and the possibility to successfully distinct the derived work from the original varies from case-to-case.

Another important difference needs to be brought out between a derivative and collective work. Essentially, collective work is defined as a work in which the work in its entirety is in

[6] D. J. Moser, C. L. Slay, *Music Copyright Law*, Course Technology PRT, (2011). ISBN: 9781435459724
[7] *Ibid*.
[8] *Ibid*.
[9] *Ibid*.

an unmodified form, along with other contributions, constitutes a separate and independent work in themselves and are assembled into a collective whole.[10] It is vital to point out that such work, which is considered to be a collective work, is not necessarily considered as derivative work. Collective works in principle consist of separate and independent copyrightable materials that have then been organised into a single unit.[11] Generally periodicals, encyclopaedias or other forms of collective pieces fall within the scope of collective works. The main common aspect for collective and derivative works lies in the fact that they are both based on pre-existing copyrightable works.[12] Initially the separation between the two can be made from their core definition, since collective work consists of a set of pre-existing works, while derivative work is a derived work based on one initial copyrighted work. However, in the complex matter of software development, the two concepts are not that clear-cut. A good way to distinguish the two in relation to software is to examine how two independent works relate to each other. A work is likely to be considered as a collective work, when two independent works are capable of sharing information, while passing such information through published interfaces or temporary connection and there has not been an a modification of one of the works by the other work.[13] However, once one of the works is modified in a manner that causes the second work to act in a unique way, the first work can be considered a derived work even though the two works, in their source code form, do not depend on each other.[14]

2.2. International Framework for Derivative Works

Berne Convention for the Protection of Literary and Artistic Works (hereinafter "Berne Convention") is one of the fundamental conventions dealing with the matter of copyright protection. There is, however, no certain reference made to derivative works. The Convention does, on the other hand, set out a certain list of uses of copyright protected works that have to be provided with a protection. Article 2(3) Berne Convention provides for the translations, adaptations, and arrangements of music and other alterations of literary as well as artistic works and stands for their protection.[15] Under the Berne Convention, in essence, the article

[10] I. Brown, *Implementing the EU Copyright Directive*, (2003).
[11] M. Webbink, *Packaging Open Source*, (2010). Available at: http://www.groklaw.net/article.php?story=20100204170037353 (Last accessed: 26.05.2014).
[12] *Ibid*.
[13] *Ibid.*
[14] *Ibid*.
[15] Berne Convention, article 2(3).

sets out that derivative works have to be protected as original works without affecting the copyright protection of the original works.[16]

Under the Agreement on Trade-Related Aspects of Intellectual Property Rights (hereinafter "TRIPS") the same provision is incorporated. Article 9 TRIPS clarifies the relation to Berne Convention and requires the members to comply with article 1 through 21 of the Berne Convention as well as the Appendix thereto.[17]

Furthermore, the World Intellectual Property Organization's (hereinafter "WIPO") Copyright Treaty has been adopted aiming to update and supplement the major existing WIPO treaties on copyright protection and in particular the Berne Convention.[18] Among other matters the purpose was to address the challenges related to the digital society in today's common picture, however the Copyright Treaty makes no clear reference to creation, distribution or exploitation of derivative works within this digital society.[19]

2.3. Derivative Works and Software Directive

The relevant directive for the harmonious interpretation of the legal aspects of computer programs is Directive 2009/24 (hereinafter "Software Directive"), however there is no exact definition given to "adaptation" or "derivative works" under the Directive. The Software Directive does on the other hand emphasise the value of interoperability. More specifically, recital 10 states that the program's function is to communicate as well as work together with other components of a computer system and with users.[20] For such purposes, a logical and appropriate physical interconnection or interaction is necessary in order to permit all elements to work with other software and hardware in all the ways in which they are intended to function.[21] It is clear that the interaction and interconnection necessity differentiates the computer programs from other forms of copyrighted works. When dealing with music or literature, no elements of interoperability occur. This is an interesting difference in relation to derivative works evolved from copyrighted work as well, since if the interconnectivity is vital for the program to operate with another software and thus modification is necessary, what is

[16] *Ibid*.
[17] TRIPS, article 9.
[18] F. Busnello, O. Romanenko, *Copyright vs. Fan-made Derivative Works*, (2012).
[19] *Ibid*.
[20] Software Directive, Recital 10.
[21] *Ibid*.

the legal status of such modification? As recital 10 states the program's function is to communicate and work together with other components of a computer system and therefore the modification in order to apply the core function could be vital. Further clarification is set in recital 15, where it is stated that an unauthorised reproduction, translation, adaption or transformation of the form of the code of the program does constitute an infringement of the exclusive rights of the proprietor.[22] In terms of "derivative works", as abovementioned, the precise wording as such is not often used in the EU legal framework, however the adaptation serves essentially the same aim. The given recital further sets out the exceptions. There can be circumstances, where the reproduction of the code of the program as well as translation of its form is vital in order to obtain the necessary information to achieve the interoperability with other programs.[23] The actual definition of interoperability is provided in recital 10, which states that it is the ability to exchange information and mutually use the information that has been exchanged.[24] Furthermore, as stated in article 6 Software Directive, decompilation of a program without the authorisation of the proprietor is allowed if it is indispensable when obtaining the information vital for achieving the interoperability of an independently created program with other programs.[25] Therefore the matter of interoperability plays a significant role when assessing the scope of adaptations allowed in order to obtain the information necessary to create the interoperability with other programs.

In regards to the scope of other adaptations, article 4 Software Directive states that it is the exclusive right of the proprietor to do or authorise the translation, adaptation, arrangement and any other alteration of the program and the reproduction of the results thereof, without prejudice to the rights of the person altering the program.[26] Therefore, it could be interpreted that when comparing the U.S. approach to derivative work and the EU legal framework, the latter has defined the term in much broader and stricter sense. Namely, it is stated that any alteration of existing programs creates a derivative work.[27] Opposed to this, the U.S. legislation creates a condition that the new work has to be based on the initial program and thus narrows the concept to an extent. As the result of the Directive's wording, one could interpret it in a way that even when a minor passage of copyrighted source code is taken into a

[22] Software Directive, Recital 15.
[23] *Ibid*.
[24] Software Directive, Recital 10.
[25] Software Directive, article 6.
[26] Software Directive, article 4.
[27] M. Välimäki, *GNU General Public License and the Ditribution of Derivative Works*, 2005 (1), The Journal of Information, Law and Technology (JILT).

combined program, this conduct may result in being considered as an "alternation of the work", whereas the overall new program might not necessarily be considered to be based on that specific code.[28] However, the given case law within the European jurisdiction indicates that the approach is somewhat similar to the U.S. understanding. In *Ibcos Computers v. Barclays Mercantile Highland Finance*, the court held that essential matter when assessing whether the copying had occurred lied in the matter of whether a substantial part of the original work had been reproduced.[29]

In addition to the economic matter of derivative works, it is necessary to also emphasise the owner's integrity, since there is a moral aspect that has to be considered in this matter as well. As stated in article 6bis of the Berne Convention, separated from the economic rights, the author also has the right to object to any modification of the given work that would be prejudicial to his honour or reputation. Therefore, in circumstances, where software is being modified to a derivative work that allegedly jeopardises the reputation or honour of the author of the initial work, the original author has the right to object the derivative and potentially stop the work from being distributed.

3. The Legal Concept of "Computer Programs"

In the early stage of the development of the computer industry, there was great uncertainty as to whether and to what extent could computer programs be protected under the copyright laws. On one hand, when written in a high-level source code the programs seemed to have all the characteristics of a creative literary work.[30] While on the other hand, computer programs could to an extent be considered as merely integral parts of operational machines.[31]

Computer programs are defined as digital instructions that are used to drive the electronic hardware of the computer itself and they may come in different forms.[32] Apart from the well-known personal computer (hereinafter "PC") model, other devices such as mobile phones, DVD players, automobile, household appliances can also embed a computer system within

[28] *Ibid*.
[29] Case *Ibcos Computers v. Barclays Mercantile Highland Finance*, (1994), FSR 275.
[30] L. B. Burgunder, *Legal Aspects of Managing Technology*, South-Western Publishing Co. : Cincinnati, (1995), p. 230.
[31] *Ibid*.
[32] W. Cornish, D. Llewelyn, T. Alpin, *Intellectual Property: Patents, Copyright, Trade Marks and Allied Rights*, 7th ed., Sweet & Maxwell Ltd : London, (2010), p. 843.

them.[33] While such embedded computing systems are designed to perform specific tasks relevant for the particular device, the computer programs to PCs are the basic operating systems of the machine and applications for a range of purposes.[34] Copyright protection has become the standard intellectual property tool in order to protect the proprietor and to prevent the copying of most types of programs.[35]

As stated in Recital 7 of Software Directive, the term of a "computer program" includes programs in any form.[36] The term also involves programs that are incorporated into hardware as well a preparatory design work that will lead to the development of an actual computer program.[37] Therefore, the definition under the Directive is relatively broad and involves programs in any form regardless of whether they are intertwined with the hardware or have a design aspect to them. Article 1 Software Directive further clarifies that for the purposes of the Software Directive, the term shall also include the preparatory design material of the actual computer programs, therefore the legal protection is also provided for the design materials as such.[38]

As indicated on the international level, article 10 of TRIPS agreement stands for the protection of computer programs regardless of whether they are in source or object code.[39] The difference between a source and object code lies in the fact that source code can be compiled into an object code. Source code is readable and thus can be modified by a man, whereas object code is a machine language consisting of the numbers of 1 and 0. Without assessing the matter in its actual depth, in principle, source code is to be containing meaningful variables and helpful comments that are intended to be read by a man and a complier then converts the source code to object code before the program can be executed.[40] Object code is a sequence of bytes that encode specific instructions for the machine that are executed by the microprocessor.[41] In legal matters however, the protection of both codes is covered in article 10 TRIPS.

---

[33] *Ibid*.
[34] *Ibid*.
[35] *Ibid.*
[36] Software Directive, recital 7.
[37] *Ibid*.
[38] Software Directive, Article 1.
[39] TRIPS, Art 10.
[40] D. Touretzky, *Source vs. Object Code: A False Dichotomy*, (2000). Available at: https://www.cs.cmu.edu/~dst/DeCSS/object-code.txt (Last accessed: 26.05.2014)
[41] *Ibid*.

Although, one could argue that the term "software" represents an umbrella definition that covers computer programs as well as the components that are required in order to run the program, for the purposes of this Thesis the two terms are used interchangeably.

4. Defying Free and Open Source Software

4.1. A Brief Overview of the Development of FOSS

The FOSS movement has a 20-year history essentially promoting the idea that software licensing agreements should allow recipients to adapt, correct and develop the given program and accompany it with other source material, subject only to a condition that such alterations to the program will be made publicly available on the same initial terms.[42] The movement has gained a major popularity over the past decade, however it does create a level of legal uncertainty specifically in relation to derivative works.

In the 1960s, when the first large-scale commercial computers were sold, they did come with some software that was free – it could be freely shared among users, it came with a source code and could be easily improved and modified.[43] The situation changed over the 1960s and by the mid-1970s, it was common to find proprietary software, that users were not allowed to redistribute, the source code was not given to the users and nothing could be modified.[44]

4.1.1. GNU Project and Linux

In 1983, motivated by a printer that could not have been fixed due to the fact that the source code had been withheld from users, Richard Stallman launched the GNU (acronym for "*GNU's not Unix!*") Project in order to write an Unix-like operating system completely free from constraints on using its source code.[45]

In regards to the GNU project, by the late 1980s, an almost complete free Unix-like operating system had been created, however the kernel was still problematic. By 1991, Linus Torvalds,

[42] W. Cornish, *Intellectual Property*, (2010), p. 845.
[43] J. M. Gonzalez-Barahona, *A Brief History of Open Source Software*, (2000). Available at: http://eu.conecta.it/paper/brief_history_open_source.html (Last accessed: 26.05.2014)
[44] *Ibid.*
[45] D. Bretthauer, *Open Source Software: A History*, (2001). Available at: http://digitalcommons.uconn.edu/cgi/viewcontent.cgi?article=1009&context=libr_pubs (Last accessed: 26.05.2014)

a Finnish university student, had released Linux kernel and it was soon licenced under the GNU GPL, since the combination of the two created a fully functional, free operating system.[46] Since Torvalds was already initially integrating GNU tools with his kernel, it has been the view of Stallman that the correct wording of the system should have been GNU-Linux instead.[47]

4.1.2. BSD Unix

Another development of the UNIX system took place in the University of California at Berkeley, where the improvement of the system led to the creation of Berkeley Software Distribution (hereinafter "BSD") Unix.[48] In the late 1980s, the Computer Science Research Group was able to distribute the system under the BSD Licence, although at that stage the users of BSD Unix still needed the AT&T initial licence, since some parts used were still proprietary.[49] In the beginning of 1990s, Bill Jolitz implemented the missing parts of the system and as a result made an unencumbered version of BSD Unix, which he named 386BSD.[50] It included utilities that enabled to form an entire operating system on a free software platform covered by only BSD licence. An important difference between the GNU GPL and the BSD licence is that the latter does not prevent the creation of proprietary software packages that are based on modified BSD code.[51]

The initial BSD licence involved an interesting advertising clause that was questioned among the community in regards to its purpose and necessity. The additional condition in the licence required that all advertising materials mentioning the features or use of this software must display the acknowledgment of the organisations involved.[52] Firstly, there is the obvious inconsistency with the later added opposite condition, which prohibits any promotion or endorsement. Secondly, in case the developers started adding code to the original work. The list of required advertising notices would then continue to increase to an unlimited degree and

[46] *Ibid.*
[47] *Ibid*.
[48] Working Group on Libre Software, *Free Software / Open Source : Information Society Opportunities for Europe?*, (2000). Available at: http://eu.conecta.it/paper.pdf (Last accessed: 26.05.2014)
[49] *Ibid.*
[50] *Ibid*.
[51] D. Bretthauer, *Open Source Software: A History*, (2001).
[52] Free Software Foundation, Inc., *The BSD Lincense Problem*, (2014). Available at: http://www.gnu.org/philosophy/bsd.html (Last accessed: 26.05.2014)

the situation would potentially become unmanageable.[53] In 1999, this condition was removed of the BSD licence and the version including the advertising restriction is not considered to be acceptable by the Open Source Initiative.[54]

4.2. The Core Aspects of FOSS

The task of specifically defying FOSS can be rather complex, since there are several distinctions, slightly different movements and variants existing in the society. However, the core idea behind the definition is rather simple.

The source code of the program is an absolute necessity in order to understand the functionality of the software, to improve and to modify it.[55] Therefore, once the access to the source code has been achieved, a skilled programmer has the possibility to work further with the software and that is the reasoning behind a major part of the developers concealing the actual source code of a developed program. The term "open source" defines something that can be modified due to the fact that its design is publicly accessible and accordingly "open source software" stands for software, which is available for modification since its source is available to the users.[56]

Therefore, the "proprietary software" or "closed source software" stands for the opposite of FOSS, since they are controlled by the person, team or undertaking that has created the program and the maintenance is exclusively controlled by an individual or a specific group of people.[57]

4.3. The Difference Between Free Software and Open Source Software

It is vital to emphasise that Stallman does not consider himself as part of the open source software movement, but instead prefers the combination of "free software".[58] Being the

[53] A. Sinclair, *Licence Profile: BSD*, (2010). Available at: http://www.ifosslr.org/ifosslr/article/view/28/62 (Last accessed: 26.05.2014)
[54] *Ibid*.
[55] Working Group on Libre Software, *Free Software / Open Source : Information Society Opportunities for Europe?*, (2000).
[56] Red Hat Inc., *What is Open Source?*, (2014). Available at: http://opensource.com/resources/what-open-source (Last accessed: 26.05.2014).
[57] *Ibid*.
[58] D. Bretthauer, *Open Source Software: A History*, (2001).

founder of the FSF, he defines free software with 4 essential freedoms: freedom to run the programs for any purpose; freedom to modify to suit one's needs; freedom to redistribute, freedom to distribute modified versions of the program.[59] The FSF's principle in this matter lies in the fact that free software is a matter of liberty and not price.[60] Free software stands for the users' freedom to run, copy, study, change, improve and distribute the software.[61] Therefore, it is vital to emphasise that the FSF and the term "free software" does not stand for software that is available for free, but rather the right to have the freedom to have access to the source code of the program and to have the possibility to edit and modify the code.

The concept of "open source" was created in 1998 in California during a strategy session after the announcement of the release of the Netscape source code.[62] It is important to note that the basic principal difference between the terms "free software" and "open software" is a rather rhetorical matter. An illustrative way of describing the two concepts could be the comparison of two political camps within the free software community, where the free software proponents stand for the more ethical freedom matters, while the "open source" supporters tend to focus more on the commercial interests.[63] For the purposes of this thesis, the term Free and Open Source Software (referred to as "FOSS") is used to bridge the gap of the two concepts.

4.4. FOSS risks

Although FOSS can be seen as a valuable way of improving technology while saving costs and increasing the vastness of development, this way of operating comes with elements of risk. The main matter emphasised in this thesis is the problematic aspect of derivative works and the lack of clarity as to when should a program be considered to be a derivative product incorporated with the open source and should fall under the same open source licence and when should the product be considered as a new original work. But the risks involved with FOSS is not only limited to the uncertainty in regards to derived works. Another potential risk

[59] *Ibid.*
[60] Free Software Foundation, Inc., *What is GNU?*, (2014). Available at: https://www.gnu.org (Last accessed: 26.05.2014).
[61] *Ibid*.
[62] Open Source Initiative, *History of the OSI*, (2014). Available at: http://opensource.org/history (Last accessed: 26.05.2014).
[63] Free Software Foundation, Inc., *Why "Free Software" is better than "Open Source"*, (2014). Available at: https://www.gnu.org/philosophy/free-software-for-freedom.html (Last accessed: 26.05.2014).

that can be associated with FOSS also include incorporating code that could infringe a patent, the possibility of violating an open source licence's attribution requirements without one's consent, while the use of FOSS also involve a lack of warranties and indemnities.[64] It is therefore not surprising that different undertakings have a level of wary in regards to using the FOSS and investing their time, money and resources in developing such software, while the legal aspects could result in a negative outcome for the undertaking itself.

Today, however, FOSS is widely used and as it has often occurred, undertakings tend to use them without always consulting with the legal department.[65] Since the use of FOSS has had an extensive increase in its use over the past years, the legal matters and the licences have developed along the process. There are more than 1400 unique licences that deal with the legal complexity of managing a FOSS in a way that would be beneficial to all parties.[66]

5. Legal Evaluation of Derivative Works in Relation to Software Programs

It can be claimed, especially in technological society, that most new works have influential elements of the existing works and thus it is vital to make the correct distinction. There is a thin line between a derivative work and an entirely new work. In a situation, where the developer only uses a minimal amount of a pre-existing work, taking only the basic functioning codes or a code that is not protected by copyright, or the pre-existing code is taken in a way that results in an substantially different program than the initial software, the works is to be considered new and original.[67]

Generally, the ownership of the a separate copyright in a derived work does not only require more than a minimal input to the prior work, but the necessity of authorisation is no less important to consider. The developer has to acquire permission from the owner of the pre-

[64] A. T. Pham, M. B. Weinstein, J. L. Ryerson, *Easy as ABC: Categorizing Open Source Lincenses*, (2010). Available at: http://www.ipo.org/wp-content/uploads/2013/03/EasyasABC.pdf (Last accessed: 26.05.2014).
[65] W. H. Venema, *Open Source Software*, National Law Journal, (2008, Oct 20).
[66] A. T. Pham, *Easy as ABC: Categorizing Open Source Lincenses*, (2010).
[67] O. Johnny, M. Miller, M. Webbink, *Copyright in Open Source Software – Understanding the Boundaries*, (2010). Available at: http://www.ifosslr.org/ifosslr/article/view/30 (Last accessed: 26.05.2014).

existing software when modifying the work and it is important to emphasise that when the permission is given, the developer of the original work will only own the copyright of the contributed work.[68] In the FOSS society, the authorisation in essence comes in the form of the FOSS licence.[69] The corresponding chapter on specific peculiarities and variations of different FOSS licences is Chapter 6.

The most problematic task in relation to derivative works is to establish the distinction between situations, where the software developer has created a derivative work from a situation, where the work could be seen as the original. The differentiation is complex and not entirely clear-cut, since the circumstances of different software programs and the input vary from case to case. The following subheadings provide the possible methods of distinguishing derived works from the original in three ways: interpretation on the basis of the source code, component based interpretation and communications based interpretation.

5.1. Source Code Approach

The source code approach in its essence compares the two source codes and the assessment is done based on that code. The legal world has mainly developed two different theories that can be applicable for the source code analyses: idea-expression dichotomy and the abstraction-filtration-comparison method.[70] This thesis will also assess the recent improvement of the latter method in the light of 5-step testing (see Chapter 5.1.1.).

The first approach, the idea-expression dichotomy is based on the understanding that copyright law does not protect ideas but only expressions of those ideas.[71] In *Nova v. Mazooma* it was stated that the well-known dichotomy is intended to apply to copyright in computer software.[72] Although the case was from the U.S, the well-known aspect indicated that this was not to be considered as an exceptional matter relevant to only some countries,

[68] *Ibid*.
[69] *Ibid*.
[70] M. Välimäki, *GNU General Public License*, (2005).
[71] N.Shemtov, *The Legal Regulation of Decompilation of Computer Programs: Excessive, Unjustified and in Need of Reform*, (2012). Available at: https://qmro.qmul.ac.uk/jspui/bitstream/123456789/3132/1/SHEMTOVTheLegal2012.pdf (Last accessed: 26.05.2014),
[72] Case *Nova Productions Limited v. Mazooma Games Limited & Others*, (2007), EWCA Civ 219, para 31.

but the link is to be considered well-known all over the world.[73] The universally held position of copyright law is that ideas, concepts, systems, procedures and methods of operation are not a subject matter to be suitable for copyright protection – this has support under TRIPS, Copyright Treaty as well as the Software Directive.[74] The concept stands for an understanding that existing programs can be studied and assessed on the basis of new original programs and the only restriction lies in the conduct of actual copying of the source or object code.[75] Essentially, the idea-expression dichotomy indicates that copyright law as such does not create restrictions on developing a new operating systems or compression algorithm, in matters where there has not been a literal copying of any of the source or object codes.

For the other approach, the abstraction-filtration-comparison (hereinafter "AFC") method, the key case lies in the case of *Computer Associtates International v. Altai*.[76] In principle, the AFC test determines the non-literal elements of the program and assesses whether these elements have been copied through a three-step procedure. *Computer Associates* had developed a program called Adapter that was used with other programs to handle the differences in operating system calls and services.[77] A former employee had gone to work for *Altai* and used the idea of Adapter to simplify the IBM operating systems there.[78] Working from Adapter listing, the employee put together a program named Oscar that was similar to Adapter, whereas about 30% of the new program came from Adapter.[79] In order to give a proper judgment to the matter, the court conducted the AFC test.

Firstly, the elements of the programs at different levels are to be abstracted. When applied to computer programs, the first step works in a manner that resembles reverse engineering on a theoretical plane, where the structure and the functions of the suspected source codes are abstracted.[80] The process starts with the code, while ending with an articulation of the program's ultimate function.[81] During the assessment at this level, it is vital to retrace and

[73] N.Shemtov, *The Legal Regulation of Decompilation*, (2012).
[74] *Ibid*.
[75] M. Välimäki, *GNU General Public License*, (2005).
[76] Case *Computer Associates International v. Altai*, (1992), 982 F.2d 693, 2d Cir.
[77] *Ibid*.
[78] *Ibid.*
[79] *Ibid*.
[80] M. Välimäki, *GNU General Public License*, (2005).
[81] L. A. Hollaar, *Chapter 2: Copyright of Computer Programs*, (2002). Available at: http://digital-law-online.info/lpdi1.0/treatise22.html (Last accessed: 26.05.2014).

map each of the steps by the designers, but in the opposite order of which they were taken when creating the program.[82]

Secondly, as given from the test name itself, is the filtration step. The aim of this level is to separate protectable elements of the expression from unprotected materials that include elements that are dictated by efficiency, external factors or elements that are taken from the public domain ([83]).[84] This step varies largely between different programs, since the necessity for filtration depends on the specific level of abstraction present.

Finally, the comparison of the protectable elements and the alleged infringement is finalised in the last step. The elements that remain after the filtration step need to be substantially similar with the elements of the program that has allegedly infringed the initial work. At this stage, the court's focus is put on the aspect of whether the defendant has copied any aspects of the protected expression as well as assessing the portion's relative importance with respect to the overall program.[85]

As abovementioned, the landmark case of *Computer Associtates International v. Altai* has its roots in the U.S Copyright system, however the same approach has been used in Europe. In a Finnish case named *Sonera Systems Oy v. VF Partner Oy*, the same AFC principle was applied in order to assess the potential infringement.[86] The test was emphasised in the opinion of a law professor and was further applied by a computer science professor as an expert witness.[87] The code had likely been copied by 10-15% and was not found to be a significant amount, however the professor stressed that the similarities can be explained since the programs were written by the same programmers, the same functionality was implemented, the user interface was similar, the programming language have only limited possibilities for variation, while the requirements forced one to implement specific parts in certain fashion.[88]

---

[82] *Ibid.*
[83] R. Nimmer, *Legal Issues in Open Source and Free Software Distribution*, (2004). Available at: http://euro.ecom.cmu.edu/program/law/08-732/Transactions/LegalIssuesNimmer.pdf (Last accessed: 26.05.2014).
[84] J. Richards, *The "Abstraction, Filtration, Comparison" Test*, (2002). Available at: http://www.ladas.com/Patents/Computer/SoftwareAndCopyright/Softwa06.html (Last accessed: 26.05.2014)
[85] L. A. Hollaar, *Chapter 2: Copyright of Computer Programs*, (2002).
[86] Case 3571, *Sonera Systems Oy v. VF Partner Oy*, (1999), R 99/661.
[87] M. Välimäki, *GNU General Public License*, (2005).
[88] I. Haikala, *Lausunto: Ohjelmistojen samankaltaisuus, VF Partner vs. Systek*, (1996). Available at:

There is however no clarity as to whether the European Courts are to apply the AFC test principle in matters related to potential infringement of computer programs.

5.1.1. POSAR testing

There has been further recent development in improving the testing in terms of specifying the legal and the judicial domain in comparison with AFC test and the new extended version is called the Planning-Operationalization-Separation-Analysis-Reporting (hereinafter "POSAR") test.[89] As the name indicates, in this approach the step-by-step process involves 5 levels instead of the 3 levels used in the AFC test.

As the first step, the POSAR test requires the abstraction of the software by separating the two software programs into their constituent structural parts by also taking into consideration the forensic importance of different factors, such as programming remarks, blunders, and errors, similarly looking items, program manuals and documents.[90] The second step involves examining and filtering the programming elements while also considering the elements from both the software packages with the goal of preparing a set of two filtered abstractions.[91] The third step deals with the separation of the suspected modifications from the two filtered abstractions, whereas the forth step covers the comparison of the remaining creative expressions of the two programs with a consideration of other contributing elements.[92] The last phase is to prepare a forensic report in a particular judiciary-friendly format.[93] When comparing POSAR test with the general AFC test, it is clear that the POSAR test deals with a more detailed process that results in an outcome that can potentially be more efficiently used in judicial procedures and can thus be a beneficial tool to assess whether a work constitutes a derivative from an original.

---

http://www.valimaki.com/org/docs/haikala_1.pdf (Last accessed: 26.05.2014).

[89] V. Bhattathiripad, *Judiciary-Friendly Forensics of Software Copyright Infringement*, (2014), Chapter 8, ISBN: 978-1-4666-5805-9.

[90] *Ibid*.

[91] *Ibid*.

[92] *Ibid*.

[93] *Ibid*.

5.2. Component Based Approach

The second approach based on the components can practically be more useable in open source development, that prefer programming paradigms relying on maximal source code reuse.[94] The component-based approach is further supported by the General Public Licence (hereinafter "GPL") separating derived works on those bases. As an example, Section 2 of version 2 states that in situations, where the identifiable sections of the work are not derived from the initial program, and can further be reasonably considered as independent from that program and can be seen as separate works, the licence at hand do not apply to those particular section when distributed as separate works. The corresponding chapter of this thesis for a specific assessment on GPL is Chapter 6.3.

There is no clear and single definition of the component in the literature. Understandably, the different understandings of specific problems and solution approaches lead to different understandings of the constituent parts.[95] While software methodologist equate components with units of project and configuration management and software architects see them as design abstractions, the legal view tends to equate components to elements that can essentially be reused.[96] The component based approach can in principle be divided into the following: components that are developer owned, components that are tailor made and third party components.[97]

A rather complex matter can potentially occur when dealing with the third party components. In a hypothetical situation, where a software developer uses a third party source code that is distributed in a component, the problem raises with the derivative status once the outcome is combined of the original main program and therefore under a partial control of the third party copyright owner.[98] From one aspect, once the third party's component is of an essential aspect for the program as a whole, one could see the potential of considering the work as being derived from such components. However, on the other hand, if the new product merely uses component's functionality and copyright law does not cover the use of interfaces, the external

[94] M. Välimäki, *GNU General Public License*, (2005).
[95] F. Bachman, *Volume II: Technical Concepts of Component-Based Software Engineering*, (2000). Available at: http://www.win.tue.nl/~mchaudro/cbse/SEI%20Technical%20Concepts%20of%20CBSE.pdf (Last accessed: 26.05.2014).
[96] *Ibid*.
[97] M. Välimäki, *GNU General Public License*, (2005).
[98] *Ibid.*

unmodified runtime component that only operates through interface could not be seen as part of the original work, thus no derivation occurs.[99] Another complex and legally uncertain situation can occur with library routines. In circumstances, where the software is linked to an external library or device drives, does this make the program a derivative work of the library? The key factor when assessing this matter lies again on the substantial role of the library, since if those routines play a significant role in the functionality of the program itself, the routine could be seen as serving a rather important role when operating with the program and could therefore be seen as an integral part of the program. Potentially, the program would not be seen as a derivative work of the server component in case it only uses the server's services, however if such program is majorly dependent on the server's functionality and could not run separately in any other setup without it, the work would likely be seen as a derived work.[100]

5.3. Communications Based Approach

Although the abovementioned approaches are essentially based on the contents of copyright law, the third approach is supported by the Free Software Foundation (hereinafter "FSF"). The FSF's approach has its grounds on more technical aspect, where the emphasis is put on the communication between the components.

When assessing whether and at what point combining two parts of components constitutes as one software program, the FSF stresses that the proper criterion depends on the mechanisms of communication and the semantics of communication.[101] This approach provides that the modules that are included in the same executable file should be seen as being combined in one program.[102] Furthermore, if the modules are designed to run in a way that they are linked in a shared address space, they shall be considered as one program.[103] In order to build a better picture of the separate versions, in case the programs are ran by other communication mechanisms that are used between two different programs (such as pipes, sockets, etc.), the programs are to be seen as entirely separate. However, if the semantics of the communication of these programs are still intense and the complex internal data structures are exchanged, that

[99] *Ibid.*
[100] *Ibid.*
[101] Free Software Foundation, Inc., *Frequently Asked Questions about Version 2 of the GNU GPL*, (2014). Availabe at: http://www.gnu.org/licenses/old-licenses/gpl-2.0-faq.html (Last accessed: 26.05.2014).
[102] *Ibid.*
[103] *Ibid.*

too could be seen as two parts of programs that form a single unit.[104] Similar to the abovementioned approaches, the key factor of the assessment lies in the significance of dependency and whether the new program can function as an individual piece. However, the approach based on communications seems to have a better link with patent law method claims criteria, since copyright law itself does not cover communication mechanisms or protocols as such.

6. FOSS Licensing and Copyleft

As abovementioned, software is protected by copyright law as literary work under the general legal norms. If one so wishes, the easiest way to put a program into the market for free, is to put it in the public domain. There would be no copyright protection on the product and users would be free to modify and distribute the software. However, the simplicity in this conduct is shadowed by the possibility for developers to convert the modified version into a proprietary work, thus gaining profit from initially free program. Therefore in order to avoid such outcomes and develop the FOSS movement, there are a great number of different licences available enabling the software developers to clarify what rights are granted to the users and what are the conditions that have to be met when operating with a FOSS.

If one has access to the source code of a certain program, he is able to study the program, modify it and further work with the program as the initial author of the software would. The concept of freedom in relation to software and its codes has a slight conflict with how the system works in practice. The fact that it is necessary to protect the free software with a licence that imposes certain restrictions has caused some controversy in certain circles, since the original concept of freedom is somehow then still restricted.[105] On the other hand, the explanation behind the necessity of licence agreements can be seen as serving the purpose of guaranteeing the perpetuation of such freedom to all of its users and supporters of this approach claim that this can only be done by limiting the ways of use and distribution of the software.[106]

---

[104] *Ibid.*
[105] Working Group on Libre Software, *Free Software / Open Source : Information Society Opportunities for Europe?*, (2000).
[106] *Ibid*.

As abovementioned, there are a great number of FOSS licences available and they differ in their characteristics. In order to get a better understanding of the FOSS licensing in practice, this thesis focuses on three licences: GPL (also briefly LGPL), BSD and EUPL. Although the authors are to provide the source code in all FOSS licence structures, the requirement for the redistributors to provide the code varies from one licence to another. Furthermore, even licences that set an obligation for the redistributor to provide the source code differ in their regulations regarding the fees for distribution. However, the core emphasis in this thesis is put on the aspect of derivative works and the legal certainty in relation to derivative works under these licence agreements. The clauses on derivative works also vary from licence to licence, since even when the access to the source code of the initial software is vital, the requirement for the access to the source code of works derived from that software is not always presented.

6.1. Copyleft Provisions

As explained, copyright law on software involves a level of legal uncertainty. However, in the context of open source software and copyleft, the vagueness is even more substantial. The licences for FOSS and for ensuring the successful existence of copyleft are in some aspects aiming the opposite goal than the regular licences in copyright law. As abovementioned, the principle purpose for FOSS licensing is to safeguard the freedom to use the software and its source code to all of the users. Therefore, one could claim that licensing is vital in order to maintain a bigger freedom, whereas the regular licensing in copyright law tends to set out specific restrictions attached to the use of the work.

The way of guaranteeing the freedom to the software is creatively called "copyleft". When the original proprietary work states "copyright, all rights reserved", the opposite version claims "copyleft, all rights reversed", "copyleft, no rights reserved" or "copyleft, some rights reserved" depending on the specificity of the licence.[107] The copyleft provisions are arguably the greatest risk posed to the commercial enterprises when engaging with FOSS.[108] The term "copyleft" – a combination of *copyright* and *left* – is used to refer to a general method or licensing scheme for making software or other work open source, while requiring all modified

[107] D. A. Frantsvog, *All Rights Reversed: A Study of Copyleft, Open-Source, And Open-Content Licensing*, (2012). Available at: http://journals.cluteonline.com/index.php/CIER/article/view/6782/6857 (Last accessed: 26.05.2014).
[108] A. T. Pham, M. B. Weinstein, J. L. Ryerson, *Easy as ABC: Categorizing Open Source Lincenses*, (2010).

versions to be open source as well.[109] Therefore, copyleft is not only about making the original work free when the copyright holder has released it, but it is also about keeping it free when further developed and distributed.[110] Therefore, in essence, what the principle of copyleft stands for in relation to computer programs, is to make the program free to be used and modified, while also providing that all derived works of the original work remain free as well.

The copyleft concept uses copyright law in order to achieve the exact opposite of the usual purpose of copyright law. Instead of aiming to privatize software, in copyleft licences the rights that are granted in order to keep the software free.[111] An author of the GPL-ed work or a copylefted work does not give up the rights as a copyright holder, but instead uses these rights in way that is different from the traditional copyright holder.[112] It is vital to emphasise that the copyleft-based licences have different effects and strengths in relation to derivative works.[113] For example, "weak-to-medium" copyleft licences are used when dealing with licensing FOSS libraries, where only changes of the FOSS library itself are subject to the copyleft provisions of the licence.[114]

While most copyleft licences are FOSS licences, not all FOSS licences are necessarily copyleft. In a situation, where the FOSS licence is not copyleft, the software that is released under such licence can be used as part of programs that are distributed under other licences.[115] One of the examples of such permissive non-copyleft licences is the BSD licence, which is further assessed in Chapter 6.4. Merely distributing a copyleft work alongside a proprietary work does not cause the latter to fall under the copyleft conditions, these provisions only apply to actual derivative works, where an existing copylefted work has been modified.[116]

---

[109] *Ibid*.
[110] S. Chen, *Free/Open Source Software Licensing*, (2006). Available at: http://akgul.bilkent.edu.tr/iosn/foss-licensing.pdf (Last accessed: 26.05.2014).
[111] *Ibid*.
[112] *Ibid*.
[113] A. T. Pham, M. B. Weinstein, J. L. Ryerson, *Easy as ABC: Categorizing Open Source Lincenses*, (2010).
[114] *Ibid*.
[115] Open Source Initiative, *Frequently Asked Questions*, (2014). Available at: http://opensource.org/faq#free-software (Last accessed: 26.05.2014).
[116] *Ibid*.

In case the author of the software wants to make the program free and simply releases the work into the public domain, the danger of the work becoming privatized and closed again is greater.[117] Therefore if the initial author aims to ensure that the work will remain freely accessible to all users, claiming the work as public domain would simply not solve the risk. By licensing the work under a FOSS licence, the author will have further assurance that the potential developments made to the software will be available to public.

6.2. Licensing Classification

As abovementioned, there are a number of different FOSS licences currently available and the conditions in them vary from licence to licence. However, in order to better observe the variations between the licences, the licence grants are categorised into three classes on the basis of their level of restrictions.

The first class stands for the non-copyleft licences that often also carry the name of "academic" or "non-viral" licences.[118] One should consider this form of licence beneficial for commercial use, since under these conditions undertakings do not have the obligation to apply the "same" licence requirements on derivative works.[119] Therefore companies have the right to choose the licence for the new work, however other non-restrictive conditions pertaining to attribution may be imposed.[120] Further, the licences may include a release of liability condition, which means that the users have to agree not to file a suit against the author of the original work or a notice of modifications requirement in case the original code has been modified.[121] One of the examples of such permissive licences is the abovementioned BSD license.

The second class and the third class can both be seen as copyleft licences with a difference between the specific conditions in terms of the constraints. Although the terms are stricter than on the non-copyleft licences, it has been pointed out by the Open Source Initiative that all FOSS licences can potentially be used for commercial purposes, however the main purpose of the software program and the likeliness of the creation of derivative works that has

[117] S. Chen, *Free/Open Source Software Licensing*, (2006).
[118] A. T. Pham, *Easy as ABC: Categorizing Open Source Lincenses*, (2010).
[119] *Ibid*.
[120] *Ibid*.
[121] *Ibid*.

to be taken into account.[122] Not all licences involving copyleft terms impose the same licence requirements on work that does not contain the code from the initial open source code and these files can be licensed under a different agreement.[123] An example of this type of licences in the GNU Lesser General Public License (hereinafter "LGPL").[124]

The strictest class of the three licensing classifications involves licence grants that require that all combined files, even those that do not contain FOSS code in any way, have to be licensed under the same licence as the FOSS work.[125] This class of "unbounded" copyleft licences considers that the derivative works are to be part of the compiled program and on the basis of this, the licence sets a requirement that any new code from the FOSS work is to be licensed under the same licence as the initial FOSS project.[126] Good examples of this type of licences are the GNU family's GPLs.[127]

According to the abovementioned classification system, it is important for the undertakings to acknowledge the variations in the FOSS licences in order to conclude a licence suitable to fit the aims of a specific project. Authors of the software have the option to use different licences for the programs, but the reality today is that the majority of the FOSS projects are covered with one of the most common licenses known, although there can be minor modifications made to the original agreement.[128] Among the possible restrictions on derivative works, there are other things that have to be considered. It is vital for the contracting party to assess the protection of moral rights in the licence, although in many legislations an extent of moral rights is protected by the legislation itself.[129] Furthermore, an important element in the licence agreement is the compatibility with proprietary licences, since some licences have formed a structure that is completely incompatible with the proprietary software and this could potentially become a problematic factor depending on the character of the project.[130] Just as

---

[122] Open Source Initiative, *Basics of Open Source*, (2014). Available at: http://opensource.org/faq#commercial (Last accessed: 26.05.2014).
[123] A. T. Pham, *Easy as ABC: Categorizing Open Source Lincenses*, (2010).
[124] Free Software Foundation, Inc., *GNU Lesser General Public License*, (2007). Available at: https://www.gnu.org/licenses/lgpl.html (Last accessed: 26.05.2014)
[125] A. T. Pham, *Easy as ABC: Categorizing Open Source Lincenses*, (2010).
[126] *Ibid*.
[127] Free Software Foundation, Inc., *GNU General Public License*, (2014). Available at: http://www.gnu.org/copyleft/gpl.html (Last accessed: 26.05.2014).
[128] Working Group on Libre Software, *Free Software / Open Source : Information Society Opportunities for Europe?*, (2000).
[129] *Ibid*.
[130] *Ibid*.

the compatibility needs to be assessed with a proprietary licence, the same has to be analysed with other FOSS licences, since there can be conflicts between the different FOSS licences as well. The latter situation occurs when the conditions of one of the FOSS licences cannot be fulfilled in case the terms of the second FOSS licence are satisfied.[131]

6.3. GNU General Public Licence

The GNU GPL has been developed over the years by the Free Software Foundation and currently the latest version of the licence agreement carries the number three. As abovementioned, the same vision of four freedoms explained by Stallman is kept in mind through the drafting process. Therefore, essentially the licence aims to guarantee that nobody is restricted by the software used.[132] The freedom to use and change the program, while also having the freedom to share either the software or the changed version of the original, still stand as the foundation of the licence agreement.[133]

Under the GPL, the copylefted software is first copyrighted, then with the distribution terms added, which in turn give legal grounds for users to have the right to use, modify, and redistribute the software program's code or any program derived from it, the software is then copylefted.[134] The code and the freedoms thus become legally inseparable, yet the condition that the distribution terms cannot be changed remains in force even if the modified work is to be distributed.

The latest version of the GNU GPL (that is version 3, thereinafter "v3") contains a basic intent of the previous version and still constitutes as a licence with a strict copyleft, although the language of the licence has been amended in order to be more comprehensive in response to technical as well as legal matters.[135] The new version aims to protect the users on three problematic issues: tivoization, laws prohibiting free software and discriminatory patent

---

[131] *Ibid*.
[132] Free Software Foundation, Inc., *A Quick Guide to GPLv3*, (2014). Available at: http://www.gnu.org/licenses/quick-guide-gplv3.html (Last accessed: 26.05.2014).
[133] *Ibid*.
[134] European Parliament, *Legal Aspects of Free and Open Source Software*, (2013). Available at: http://www.europarl.europa.eu/document/activities/cont/201307/20130702ATT68998/20130702ATT68998EN.pdf (Last accessed: 26.05.2014).
[135] Institute for Legal Questions on Free and Open Source Software, *What is the Difference between GPLv2 and GPLv3?*, (2013). Available at: http://www.ifross.org/en/what-difference-between-gplv2-and-gplv3 (Last accessed: 26.05.2014)

deals.[136] First matter stands for a situation, where the system does corporate a GPL-ed software, but the hardware restrictions are used such a way that prevent users to run modified version of the given software on that particular hardware.[137]

The second matter touches acutely upon the EU legislation, namely the Directive 2001/29/EC (hereinafter "InfoSoc Directive"). The legislation does not allow to write or share any software that can break Digital Restriction Management (hereinafter "DRM").[138] DRM is defined as any technology that is created into an electronic product or service and aims to limit its range of uses after the purchase.[139] The technology works in a way that prevents customers from using the technology in such ways that do not meet the business agenda of the content provider.[140] It is an issue that has both supporters and opponents, while the latter stand for the vision of users having the right to unlimited use once the product has been purchased. On the European level the issue raises with the InfoSoc Directive, where article 6 stands for the necessity for member states to provide adequate legal protection against circumvention. The article stresses that the protection is needed against any circumvention of effective technological measures and against any activities (manufacturing, importing, distributing, selling etc.) that essentially promote, advertise or market circumvention.[141] In principle, GNU is of the opinion that such legislation should not intervene with the rights acquired by the GPL and thus some necessary amendments to the GPL v3 were made.

Thirdly, the issue occurred with discriminatory patent deals. It was stated by Microsoft that no suing of free software users would occur as long as the software is obtained from a vendor who is paying Microsoft for the privilege.[142] In other words, Microsoft aimed to get royalties for the use of free software under conditions that are conflicting with the GNU's principles and thus the solution for this was further developed in the new version.

Furthermore, the new version addresses questions that were insufficiently covered in the previous version, such as the issue of compatibility, which in essence now simplifies

[136] Free Software Foundation, Inc., *A Quick Guide to GPLv3*, (2014).
[137] *Ibid*.
[138] *Ibid*.
[139] B. Mruk, M. Lee, *What is DRM?*, (2012). Available at: http://drm.info/en/what-is-drm (Last accessed: 26.05.2014).
[140] *Ibid*.
[141] InfoSoc Directive, article 6(1)&(2).
[142] Free Software Foundation, Inc., *A Quick Guide to GPLv3*, (2014).

combining GPL code with code that has been published under a different licence.[143] Furthermore, the regulations in relation to digital rights management were applied in order to keep GPL software from being changed at will since users appealed to the legal regulations to be protected by "TPMs" (technical protective measures).[144] An important note to stress here is the fact that, the European Commission had been invited to participate in the making of the GPL v3, but the Commission declined on the grounds that the governments participating were not of the Commission's level of importance and dignity.[145]

Therefore, in essence, what the GNU GPL stands for is that when developers write software and release it under the terms of GPL, it will be free software and will remain free software regardless of who modifies or distributes the software. Therefore, the core aim is to set aside the restrictive measures provided by the proprietary software and guarantee freedom to use the software while also contributing to the technological society yourself.

6.3.1. Derivative Works

With the GNU GPL concept, there is no limitation for the derived works when only used internally, but copyleft is to be applied when derivative works are distributed in public and the licence sets out that the derived works in such circumstances have to be as free as the original work.[146] By licensing under the GNU GPL, the initial authors will set out a requirement for the people distributing the program and for the developers that distribute modified works, to have the obligation to keep the derivative works as free as the original work.[147]

Therefore, the source code of a software program is thus openly distributed, which potentially encourages developers to study the code, get inspired and hopefully contribute more to the society. However, from another angle, the licence also creates a level of limitation in distribution. GNU GPL strictly controls the further distribution of such derivative works, since it requires that the licence terms of the derivative works are not to be changed.[148] The core aim behind this restriction lies in the principle that since the original developer of the

[143] Institute for Legal Questions on Free and Open Source Software, *What is the Difference between GPLv2 and GPLv3?*, (2013).
[144] *Ibid*.
[145] European Parliament, *Legal Aspects of Free and Open Source Software*, (2013).
[146] *Ibid*.
[147] S. Chen, *Free/Open Source Software Licensing*, (2006).
[148] M. Välimäki, *GNU General Public License and the Ditribution of Derivative Works*, 2005.

software has provided the source code for the general public, it would be a logical implication that a subsequent developer should also contribute to the public and in order to ensure this, the derivative work has to follow the same licence conditions.[149] One can see the restriction created by this condition, however as the aim of the GNU's project consists in the matter of liberty, the necessity to ensure that the derivative works fulfil the same goal is intelligible. As GNU has described this approach in the preamble of the licence, the condition is there in order to respect the freedom of others and thus certain responsibilities occur if one is to distribute copies of the software.[150] Developers who use the GPL protect the rights with two steps: firstly, they assert copyright on the software and secondly, they offer the licence giving one the legal permission to copy, distribute and modify the particular piece of work.[151]

There is further a possibility that a developer has created an aggregate, where other conditions apply. Essentially, an aggregate embodies a number of programs that are separate, but distributed on the same media.[152] Under the GPL conditions, it is possible to create and distribute an aggregate even under circumstances, where the other software programs involved are non-free or GPL-incompatible, yet the only requirement that has to be met, is that the developer cannot release such aggregate under a licence that would deny the users to exercise the rights granted by each program's individual licence.[153] However, the matter of when a programs constitutes as a single unity or can be seen as two individual programs is still problematic and need to be assessed on case-by-case basis.

Theoretically, the issue of derivative works does not come into play under conditions, where the programs are to be considered as separate works and the influence of one of the programs to another is non-existent. However, it is vital to understand the principle distinction between the possibility that programs have created a single unit and a situation, where the software programs are to be seen separate. GNU is of the opinion that the distinction depends both on the communication mechanism as well as the semantics of the communication.[154] Therefore, the first assessment should be done with analysing the mechanism that are used in order to

[149] *Ibid*.
[150] Free Software Foundation, *GNU General Public License*, (2007).
[151] *Ibid*.
[152] Free Software Foundation, Inc., *Frequently Asked Questions about the GNU Licenses*, (2014). Available at: http://www.gnu.org/licenses/gpl-faq.html#MereAggregation (Last accessed: 26.05.2014).
[153] *Ibid*.
[154] *Ibid*.

link the programs, while the second matter involves the substantial elements indicating what information is changed between the programs.

6.3.2. Lesser General Public Licence

The general concept of the LGPL is to permit modification as well as distribution of free derivative works, but it precludes the creation of proprietary derivative works. As abovementioned, not all licences impose the same requirements on work that does not involve the code form the initial source and such files can successfully be licensed under a different agreement – LGPL is an example of that.

The core difference between the GPL and LGPL is that the latter allows the use of the library in proprietary software, while the GPL for a library enables it only for free programs.[155] Therefore, the developer needs to particularly set out the aims for each individual program and decide whether there occurs the necessity to use the library in proprietary software, when choosing the appropriate licence agreement. Mostly, the LGPL can be a reasonable choice when a free library's features are available for proprietary software through other alternative libraries, since the library cannot provide any additional advantage to the free software.[156]

Section 2 of the LGPL provides for the distribution of the modifications. It states that if a developer modifies the copy of the library and in those modifications a facility is referring to a function or data that is to be supplied by an application that uses the facility, then in principle such copy of the modified version can be conveyed under specific conditions.[157] The requirement is either to convey the version under the same LGPL licence, while ensuring that the facility still operates in the event that the application does not supply any function or data, or under the GNU GPL with none of the additional consent of the LGPL applicable.[158]

6.4. BSD Licences

When considering popularity by the frequency of use, the BSD licence has been ranking at the top of the list after the GNU GPL family and therefore, for the purposes of giving an accurate

[155] Free Software Foundation, Inc., *Why You Shouldn't Use the Lesser GPL for Your Next Library*, (2014). Availablt at: http://www.gnu.org/licenses/why-not-lgpl.html (Last accessed: 26.05.2014).
[156] *Ibid*.
[157] LGPL, section 2.
[158] *Ibid*.

overview on the essential issues of this thesis, it is vital to assess the peculiarities regarding the BSD and the conditions of derivative works in relation to the BSD licence. According to the analyses in 2012, approximately 7% of the FOSS projects use some form of BSD licence, which made it the third most popular licence next to the GPL licences.[159]

The core of the BSD agreement is simplicity. The licence has a three-part structure that sets forth an elementary copyright notice, has a short licence grant and warranty disclaimer with a limitation of the liability clause.[160] The BSD licence is particular due its refreshing shortness and the fact that the entire licence can fit onto one page of paper.[161] Essentially, what the BDS licences cover is the right to use, copy and distribute unmodified and modified source or binary forms of the program, while all the distributed programs were to be accompanied by the licence.[162] The names of the previous contributors are not to be used in order to promote any modified versions without their acknowledgment and written consent.[163]

The first part of the three-level structure covers the copyright notice and in principle, follows the style of the traditional copyright notice.[164] Copyright notices are a common practice, since they still serve the purpose of identifying the copyright owner to recipients of the created work and given the fact that the BSD licence was initially developed in the 1980s, the notice was then still required for enforceability under the US Copyright Act.[165] Another interesting aspect to the BSD licence is the still remaining "all rights reserved" notice that is not commonly used among other FOSS licences. Although what seems to be a conflicting matter with the aim of the FOSS movement, it is important to emphasise that not all the rights are reserved in this aspect as the author is granting many rights in the same instrument as the notice, but one could indeed see the BSD licence as being one of the more narrow licence grants.[166]

---

[159] R. Wilson, *The Modified BSD License – An Overview*, (2012). Available at: http://oss-watch.ac.uk/resources/modbsd (Last accessed: 26.05.2014)
[160] A. Sinclair, *Licence Profile: BSD*, (2010).
[161] R. Wilson, *The Modified BSD License – An Overview*, (2012).
[162] *Ibid*.
[163] *Ibid*.
[164] A. Sinclair, *Licence Profile: BSD*, (2010).
[165] *Ibid*.
[166] *Ibid*.

When assessing the second part of the BSD licence, it is interesting to note that the heart of the licence agreement is formed in a single sentence. As abovementioned, the BSD licence stands for the right of redistribution and use in source or binary forms, with or without modification, in circumstances, where the necessary conditions are fulfilled.[167] Firstly, the redistribution of the source code has to retain the copyright notice, the list of conditions and the disclaimer, while the redistributions of the binary forms have to reproduce the copyright notice, the conditions and the disclaimer in the documentation or other materials that are provided with such distribution.[168] There are two versions of the BSD licences, subsequently named as the "2-clause licence" and the "3-clause licence". The difference between the two versions lies in the optional third condition that can be added to the agreement. The third condition stands for the requirement that neither the name of the potential undertaking nor the names of the contributors may be used in order to endorse or promote products that are derived from the initial software without their written consent.[169]

As software licensed under the BSD licence does not necessarily involve a charge or a loyalty, it is reasonable that the licensees of the program do not receive commercial guarantees.[170] The final part of the BSD licence stands for this. Moreover, considering that fact that the distribution stream would make warranties and liability terms difficult to implement, the BSD licence wisely applies this to all the upstream copyright holders and contributors.[171] Therefore, the licensee does have the right to use and redistribute the modified or unmodified program, but no warranty is given with the licence and none of the initial authors of the software have any liability in regards to the merchantability or fitness of the program.[172]

6.4.1. Derivative Works

Although there is no explicit clause in the BSD licence granting the licensee a right to modify the original software, the fact that the licence includes the condition "with or without

---

[167] Open Source Initiative, *The BSD 2-Clause License*, (2014). Available at: http://opensource.org/licenses/bsd-license.php (Last accessed: 26.05.2014).
[168] *Ibid*.
[169] Open Source Initiative, *The BSD 3-Clause License*, (2014). Available at: http://opensource.org/licenses/BSD-3-Clause (Last accessed: 26.05.2014).
[170] A. Sinclair, *Licence Profile: BSD*, (2010).
[171] *Ibid*.
[172] B. Montague, *Why You Should Use a BSD Style License For Your Open Source Project?*, (2013). Available at: https://www.freebsd.org/doc/en/articles/bsdl-gpl/article.html#bsd-advantages (Last accessed: 26.05.2014)

modification" gives grounds to a reasonable presumption that the licence does not preclude the right to modify the software.[173] Furthermore, when dealing with the 3-Clause licence, the additional condition specifically mentions the possible derivations of the original product giving further support to the understanding that derived works of the initial software are in accordance with the purposes of the given licence. This clause successfully eliminates the potential threat of jeopardizing the honour or reputation of the initial author as set out in article 6bis of the Berne Convention, since the linkage with the original contributors is detached.

As it is clear that the licensee does have the right to distribute the work in accordance with the BSD licence agreement, it would be difficult to argue that the licensee does not also have the right to modify the program.[174] As further indicated, when the permissive licences implement no conditions for copying or merging the covered code, the interoperability in this sense is of no issue.[175] It is necessary in this aspect to emphasize that the term "interoperability" can be used in relation to software programs' co-operating abilities, while in the context of licences the legal term stands for the possibility to reuse the covered code in other projects in combination with codes that are covered with other licences, while keeping the freedom to distribute the resulting combination.[176] This term also includes works that could potentially be considered as derivative works under the copyright law.[177] While the BSD licence and similar permissive licences are generally considered to be compatible with the copyleft licences, there is an element of uncertainty in regards to the legal effect of combining the codes under the BSD licence with the codes under the copyleft licence.[178] Since the BSD licence does not explicitly include the right to sublicense, there is an option that the BSD licence would be compatible since the combined code can then be re-licensed under the copyleft licence, but in those circumstances the reliance is then based on the licensor's intent and the interpretation of the community to read the sublicense right into the BSD licence's terms.[179] However, it is vital to focus on the fact that the core of the open source modes is a direct grant form the

[173] A. Sinclair, *Licence Profile: BSD*, (2010).
[174] *Ibid*.
[175] European Parliament, *Legal Aspects of Free and Open Source Software*, (2013).
[176] *Ibid.*
[177] *Ibid*.
[178] Free Software Foundation, Inc., *Various Licenses and Comments About Them*, (2012). Available at: https://www.gnu.org/licenses/license-list.html#GPLCompatibleLicenses (Last accessed: 26.05.2014).
[179] A. Sinclair, *Licence Profile: BSD*, (2010).

copyright owner to the licensee and this does not involve sublicenses.[180] In the opposite situation, where instead of a sublicense, the combined codes would continue to be licensed under the BSD licence, a conflict would occur with the terms of the copyleft licence, since the latter traditionally requires that the derivative works would be licensed under the copyleft licence.[181] Therefore, the compatibility matters are not clear and do have potential to give grounds on possible conflicts between the different licensing grants. The situation again varies on the specific circumstances and peculiarities of the program, which, to a degree, further explains the vagueness in this matter at hand. In principle, however, the general community interpretation seems to claim that fundamentally the two versions of licences are to be considered as being compatible.[182]

Although there is arguably an element of legal obscurity in BSD licences, the simplicity and shortness of the licence grant has made BSD licences one of the most popular FOSS licences in the technological society. The fact that the licence's language includes some clues in regards to the rights that are to be assumed to accompany the grant further supports that it is indeed a very permissive licence.[183] As abovementioned, the BSD licence does not include a specific clause giving the right to modify the codes, however this approach can to an extent be implied. Furthermore, another right that has not been specifically brought out, is the right to reproduce. Some right of reproduction can potentially be read into the right to modify, since it would be impractical to state that as the license covers computer code and gives the right to the licensee to modify and distribute the software, but does not permit reproducing of the software.[184]

6.5. European Union Public Licence

While the majority of the FOSS licences has been developed in the United States and subsequently have had the impact of the copyright law within the United States, the European Union has also developed its own template. The European Union Public Licence (hereinafter "EUPL"), published by the Commission, was first launched in 2007.[185] In 2012, the licence

[180] *Ibid*.
[181] A. Sinclair, *Licence Profile: BSD*, (2010).
[182] Free Software Foundation, *Various Licenses and Comments About Them*, (2012).
[183] A. Sinclair, *Licence Profile: BSD*, (2010).
[184] *Ibid*.
[185] European Parliament, *Legal Aspects of Free and Open Source Software*, (2013).

was used for more than 500 software and non-software projects in the Union.[186] When drafting the EUPL, certain requirements were set, while the aim was for the Commission to lead by example of distributing its own produced software in order to encourage the public sector in the member states to do the same.[187] The EUPL is, similarly to GPL, a copyleft licence that aims to avoid exclusive appropriation of the software and, although initially aimed to focus on the public sector, the licence can be used by the EU institutions, member states as well as individuals.[188]

When drafting the licence, there were specific requirements that the Commission underlined considering the peculiarities of the Union's operation. Among the core principle of providing FOSS freedoms and ensuring the protection from exclusive software appropriation, the Commission also stressed the fact that the EUPL would have a working value in all official EU languages so that there would not be a necessity for sworn translators in the court and other institutions for translations.[189] Furthermore, it is the essential condition for the EUPL to be in accordance with the European copyright legislation and terminology, while clarifying the applicable law and competent courts as requested by the EU institutions.[190] The EUPL is generally considered relatively comprehensive and not too complex, although the length of the licence exceeds the short versions of BSD licences.

6.5.1. EUPL and Derivative Works

As abovementioned, the EUPL is to be considered a copyleft licence. Therefore, in principle EUPL provides for making the software free and further requires that the modified and derived versions of the program remain to be free as well. The legal instrument is formed in a way that the it gives users the right to use, modify and redistribute the software program's code as well as any derived work only with the condition that the terms are not changed.

As state by article 2 of the EUPL, the users have the right to modify the original work as well as to make derivative works from the original.[191] One has the right to sell the software or works covered by the EUPL as well as related services at a determined price, but once that is

[186] *Ibid*.
[187] P. Schmitz, *The European Public Licence (EUPL)*, (2013), International Free and Open Source Software Law Review, vol. 5 issue 2.
[188] European Parliament, *Legal Aspects of Free and Open Source Software*, (2013).
[189] P. Schmitz, *The European Public Licence (EUPL)*, (2013).
[190] *Ibid*.
[191] EUPL, article 2.

done, the covered work cannot be further subject to the management of royalties.[192] This approach constitutes a fundamental principle to FOSS licensing and there is a reasonable elucidation behind this requirement. Namely, when freedom would be granted to all potential recipients to exploit and further develop derivative works and to in turn distribute these works, the control of the use for the royalty charging would in reality become impossible to implement.[193]

For the matters of interoperability, article 5 of the EUPL is of relevance. It states that in case the licensee distributes or communicates derivative works that can be based upon both the original work licensed under the EUPL as well as another work that is licensed under another compatible licence, it is possible for the conduct to be done under the terms of that compatible licence. The conditions for the variable copyleft principle requires that the software code is covered by the EUPL and has been combined in or with another work.[194] Further, it is required that such a combination forms a derivative work, while the merged code is to be licensed as a whole and distinct licences under the given conditions is not allowed.[195] The other work merged with the code previously covered by the EUPL has to be obtained under a compatible licence from a given list by the Commission, whereas then the same compatible licence can be used in order to protect the derived work as a whole.[196]

7. Conclusive Legal Analysis on the FOSS Licences and Derivative Works

As can be derived from the previous assessment on different FOSS licences, there are a number of aspects that need to be considered when choosing the right licence for a specific program.

The keyword for the BSD licences is simplicity – it is short and permissive. The agreement's main focus is put on the fact that the author of the program bears no liability in any event regardless of whether a direct or indirect damage to the user has occurred. In principle, when

[192] European Parliament, *Legal Aspects of Free and Open Source Software*, (2013).
[193] P. Schmitz, *The European Public Licence (EUPL)*, (2013).
[194] European Parliament, *Legal Aspects of Free and Open Source Software*, (2013).
[195] *Ibid*.
[196] *Ibid*.

using the BSD licence, the software itself is as free as a GPL-ed software, but the essential difference lies in the conditions set out in relation to derivative works. The reasoning behind the sceptical approach in relation to the BSD licences lies in the fact that an undertaking can potentially take a BSD licenced code and incorporate it into its own proprietary work. Therefore, in circumstances, where the distribution of derived works is of importance, the BSD licence may not be the best option. Since this licence constitutes a non-copyleft licence, the matter of differentiating between original work and derivative work is of little significance, since the copyright owner has not set restrictive conditions on the original work itself.

However, when dealing with the GPL or EUPL, the conditions involving derivative works are rather different. One of the main differences between the GPL and the EUPL is the fact that while the GPL has a rather restrictive approach on the possibility of changing for another licence, the EUPL provides a compatibility test. When it is necessary in order to avoid licence conflicts, the software developers have the possibility to license a composed work under a similar copyleft licence and are not necessarily bound to apply the EUPL.[197] As abovementioned, the GPL stands for the four freedoms, while providing full support to the copyleft principle. The core idea with the copyleft licences is to avoid the possibility that the users would be restricted by the software they use. However, in order to achieve this aim, the legal inseparability is created through the licence requirements. Namely, the goal is achieved through the condition that the distribution terms cannot be changed when distributing the derivative works. With this concept, it is vital for both parties to have a clear understanding as to what constitutes a new work and what can still fall under the scope of being a derived work from a GPL-licensed product. As there are currently no case law or disputes in Europe that would involve the interpretation of the GPL or EUPL and no case law concerning the concept of software adaptation, the clarity in the actual assessment process is still not certain in its entirety.

When assessing the matter specifically in the context of EU legislation, the case of *SAS Institute* is of relevance. It is clear that the basis of the software program is to be assessed when considering whether the work constitutes as a derivative. In *SAS Institute*, the case dealt

[197] P. Schmitz, *EUPL or GPLv3? A Comparison Table of the Main Characteristics and Differences*, (2009). Available at: https://joinup.ec.europa.eu/community/eupl/news/eupl-or-gplv3-comparison-table-main-characteristics-and-differences (Last accessed: 26.05.2014).

with two undertakings – SAS Institute is a developer of analytical software, while World Programming Ltd. (hereinafter "WPL") developed an alternative product designed to emulate the SAS components as closely as possible and the attempt was to ensure that the same inputs would produce the same outputs.[198] This would further enable the users of the SAS system to run the developed scripts initially used with the SAS in WPL's system. The referring court emphasised that it had not been established, that in order to develop the system, WPL had had access to the source code of the SAS system, that it had copied any of the text of that source code or had copied any of the structural design of the source code.[199] Although this statement essentially eliminates the possibility of derivative works, since the conduct of fully developing the system without having access to the initial program's code is implied, there are still some vital points to bring out from this ruling. Although Directive 91/250/EEC was applied to this ruling, the same conditions can be found in the more recent Software Directive. Article 1(2) of both Directives provides that the protection of the Directive applies to the expression in any form of a computer program, whereas the ideas and principles are not protected.[200] The Court of Justice of the European Union (hereinafter "CJEU") clarified that neither the functionality of the program nor the programming language and the format of data files used in order to exploit certain aspects of its function constitute a form of expression and are therefore not protected by copyright.[201] When developing the ruling's tendency, the Court seems to be rather concise in its view that the only protection provided by copyright law is the protection of the expression of the idea. One can potentially argue, that such approach gives propensity towards the method of idea-expression dichotomy, since it is also based on the core understanding that copyright law protects merely the expression of ideas and does not provide protection on such ideas themselves. This approach has further support from the TRIPS agreement, it is commonly stated that the ideas, concepts, systems, procedures and methods of operation are not a subject matter to be suitable for copyright protection. When following this pattern, the essence of a possible derivative work thus rests fully on the source code. Therefore, when drawing a parallel with the ruling in *SAS Institute* while putting it in the context of FOSS licensing, the only way to fall under the derivative work requirements is thus when there has been a literal copying of the code.

[198] Case C-406/10, *SAS Institute Inc. v World Programming Ltd.*, (2012).
[199] *Ibid*.
[200] Software Directive, Directive 91/250/EEC.
[201] Case C-406/10, *SAS Institute Inc. v World Programming Ltd.*, (2012)

However, the GPL tends to support the component-based approach when distinguishing whether the work constitutes a derivative work. Once the third party's component is of an essential element for the new program as a whole, the work is to be considered as derived from the initial software component. Although, it is important to stress that in case the new product merely uses component's functionality and since copyright law does not cover the use of interfaces, an external unmodified component that operates through an interface shall not be seen as a substantial part of the work, and thus no derivation would occur. The general approach supports the understanding that communication over interfaces does not create a derivative work as such. The third option is to assess the matter form more complex technical aspects, while focusing on the communication of the components and the similarities from this perspective. As stressed by the FSF, the proper assessment shall depend on the mechanisms and the semantics of communication. The key factor in this approach lies in the level of dependency and whether the new product is capable of functioning as an individual piece.

7.1. Hypothetical Example on Modifying FOSS Licensed Source Code

Hypothetically, a software developer "A" has in his possession a copy of a program "P" that is licensed under the GPL principles. Next, A is entitled to modify the program by developing some specific elements of the code, while also having the right to delete some functions from the code. In doing so, a new program "N" is created. Considering the fact that P was protected by the GPL, the issue of whether the new program N is also subject to the GPL conditions. An interesting fact to accentuate is that, if A would only aim to use the new product privately, there would be no necessity for applying GPL, the condition only occurs when the product is being distributed. Therefore, depending on the aim of the developer, the derivative work may never have to meet the requirements of the GPL. However, in case developer A wishes to exercise the right to distribution of the derivative work, the new product needs to be subjected to the term of GPL.

Further, if A wishes to distribute the product, it is important to ensure that the work is indeed a derived work from the original and that the modifications to the initial source code were not as substantial to create a completely new product. Considering the hypothetical facts of the case and the abovementioned possibilities on differentiating the derivative work from the original, it is rather clear that product N would constitute a derivative of the product P. Firstly, for a work to constitute a derivative of another work, it has to borrow from the latter and this

conduct has occurred in the present example. Secondly, the work has to adapt the work upon which the new work is based, thus when developing program P in this extent, the developer has created a derivative and under the given circumstances, has to meet the terms of the GPL.

8. Conclusion

In essence, derivative works or as often referred to within European legislation – adaptations, are works that have been based upon a pre-existing copyrighted work or works. In principle, both EU and U.S. recognise derivative work rights as part of copyright that include the right to alter content of an initial work, take extracts from the original work, combine them, translate them, or otherwise create a new work from the existing work, an owner of the creative work does have the absolute right to create such derivative works. There are two conditions that have to be met in order to fall within the scope of the definition. Firstly, it is vital that the derived work has borrowed form the pre-existing work and secondly, the work has to recast, transform or adapt the pre-existing work. Although the core of the definition is to an extent understandable, the problematic uncertainty occurs in the actual extent of originality required in order to create a fully separate work.

In matters of software programming, the essential question is whether it is purely the programming architecture that determines if a work constitutes a derivative from a pre-existing work. This thesis gives guidance on the potential approach of defying a derivative work in software matters, indicating that it is not purely a comparison of the two source codes that brings the possible result, as the function of the source code can still be similar even if the code functioning language has been consciously modified. Alternative ways of defying whether the work could be seen as a derivative involve component-based as well as communication-based approaches. There are also a number of tests, such as the AFC test and POSAR test, available in order to compare the works. However, since the CJEU has not so far given a ruling concerning the concept of adaptation of software, the certainty in the matter is yet to be established.

Essentially, software is protected by copyright law as literary work. The easiest way to put a program into the market for free is to put it in the public domain. Under these conditions, there would be no copyright protection on the product and users would be free to modify and distribute the software. However, the simplicity in this conduct is shadowed by the possibility

for developers to convert the modified version into a proprietary work, thus gaining profit from a program that was initially free. In order to avoid such outcomes as well as to further develop the FOSS movement, there are a number of different licences available enabling the software developers to clarify what rights are granted to the users and what are the conditions that have to be met when operating with a FOSS.

With the FOSS licences, a new definition of copyleft has developed. The copyleft concept uses copyright law in order to achieve the exact opposite of the usual purpose of copyright law. Instead of aiming to privatise software, in copyleft licences the rights that are granted in order to keep the software free. An author of the copylefted work does not give up the rights as a copyright holder, but instead uses these rights in way that differ from the traditional copyright holder. The copyleft-based licences have different effects and strengths in relation to derivative works and while most copyleft licences are FOSS licences, not all FOSS licences are automatically copyleft.

Unfortunate for most software developers, creating a clear cut rule for establishing when a work would constitute a derivative work of the pre-existing software program is not possible, since the matter is complex and requires a case-by-case assessment. Essentially, if a code has been copied from an existing code protected by a FOSS licence and minor revisions has been made to it, the work constitutes a derivative work. In case the code has been written by a developer itself and perhaps the idea behind a pre-existing work has been an element of motivation, then the work should not be seen as a derivative of the pre-existing work. However, scenarios between the two simple concepts require a case-by-case legal analysis.

Bibliography

OFFICIAL MATERIALS

Berne Convention for the Protection of Literary and Artistic Works

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

Directive 2009/24/EC of the European Parliament and of the Council of 23 April 2009 on the protection of computer programs

Agreement on Trade-Related Aspects of Intellectual Property Rights

U.S Code Title 17 Chapter 1 § 101.

BIBLIOGRAPHY

F. Bachman, *Volume II: Technical Concepts of Component-Based Software Engineering*, (2000). Available at: http://www.win.tue.nl/~mchaudro/cbse/SEI%20Technical%20Concepts%20of%20CBSE.pdf

V. Bhattathiripad, *Judiciary-Friendly Forensics of Software Copyright Infringement*, (2014), ISBN: 978-1-4666-5805-9.

D. Bretthauer, *Open Source Software: A History*, (2001). Available at: http://digitalcommons.uconn.edu/cgi/viewcontent.cgi?article=1009&context=libr_pubs

I. Brown, *Implementing the EU Copyright Directive*, (2003). Available at: http://www.fipr.org/copyright/guide/eucd-guide.pdf'

L. B. Burgunder, *Legal Aspects of Managing Technology*, South-Western Publishing Co. : Cincinnati, (1995).

F. Busnello, O. Romanenko, *Copyright vs. Fan-made Derivative Works: Unresolvable Conflict or Unavoidable Impulse for Reform of the Present Copyright System*, (2012). Available at: http://www.academia.edu/1255504/Copyright_vs._Fan-made_Derivative_Works_unresolvable_conflict_or_unavoidable_impulse_for_reform_of_the_present_copyright_system

S. Chen, *Free/Open Source Software Licensing*, (2006). Available at: http://akgul.bilkent.edu.tr/iosn/foss-licensing.pdf

W. Cornish, D. Llewelyn, T. Alpin, *Intellectual Property: Patents, Copyright, Trade Marks and Allied Rights*, 7th ed., Sweet & Maxwell Ltd : London, (2010).

European Parliament, *Legal Aspects of Free and Open Source Software*, (2013). Available at: http://www.europarl.europa.eu/document/activities/cont/201307/20130702ATT68998/20130702ATT68998EN.pdf

D. A. Frantsvog, *All Rights Reversed: A Study of Copyleft, Open-Source, And Open-Content Licensing*, (2012). Available at: http://journals.cluteonline.com/index.php/CIER/article/view/6782/6857

Free Software Foundation, Inc., *What is GNU?*, (2014). Available at: https://www.gnu.org

Free Software Foundation, Inc., *The BSD Lincense Problem*, (2014). Available at: http://www.gnu.org/philosophy/bsd.html

Free Software Foundation, Inc., *Why "Free Software" is better than "Open Source"*, (2014). Available at: https://www.gnu.org/philosophy/free-software-for-freedom.html

Free Software Foundation, Inc., *Frequently Asked Questions about Version 2 of the GNU GPL*, (2014). Availabe at: http://www.gnu.org/licenses/old-licenses/gpl-2.0-faq.html'

Free Software Foundation, Inc., *Frequently Asked Questions about the GNU Licenses*, (2014). Available at: http://www.gnu.org/licenses/gpl-faq.html#MereAggregation

Free Software Foundation, Inc., *GNU Lesser General Public License*, (2007). Available at: https://www.gnu.org/licenses/lgpl.html

Free Software Foundation, Inc., *GNU General Public License*, (2014). Available at: http://www.gnu.org/copyleft/gpl.html

Free Software Foundation, Inc., *A Quick Guide to GPLv3*, (2014). Available at: http://www.gnu.org/licenses/quick-guide-gplv3.html

Free Software Foundation, Inc., *Why You Shouldn't Use the Lesser GPL for Your Next Library*, (2014). Availablt at: http://www.gnu.org/licenses/why-not-lgpl.html

Free Software Foundation, Inc., *Various Licenses and Comments About Them*, (2012). Available at: https://www.gnu.org/licenses/license-list.html#GPLCompatibleLicenses

J. M. Gonzalez-Barahona, *A Brief History of Open Source Software*, (2000). Available at: http://eu.conecta.it/paper/brief_history_open_source.html

I. Haikala, *Lausunto: Ohjelmistojen samankaltaisuus, VF Partner vs. Systek*, (1996). Available at: http://www.valimaki.com/org/docs/haikala_1.pdf

L. A. Hollaar, *Chapter 2: Copyright of Computer Programs*, (2002). Available at: http://digital-law-online.info/lpdi1.0/treatise22.html

Institute for Legal Questions on Free and Open Source Software, *What is the Difference between GPLv2 and GPLv3?*, (2013). Available at: http://www.ifross.org/en/what-difference-between-gplv2-and-gplv3

O. Johnny, M. Miller, M. Webbink, *Copyright in Open Source Software – Understanding the Boundaries*, (2010). Available at: http://www.ifosslr.org/ifosslr/article/view/30

B. Montague, *Why You Should Use a BSD Style License For Your Open Source Project?*, (2013). Available at: https://www.freebsd.org/doc/en/articles/bsdl-gpl/article.html#bsd-advantages

D. J. Moser, C. L. Slay, *Music Copyright Law*, Course Technology PRT, (2011). ISBN: 9781435459724

B. Mruk, M. Lee, *What is DRM?*, (2012). Available at: http://drm.info/en/what-is-drm

R. Nimmer, *Legal Issues in Open Source and Free Software Distribution*, (2004). Available at: http://euro.ecom.cmu.edu/program/law/08-732/Transactions/LegalIssuesNimmer.pdf

Open Source Initiative, *History of the OSI*, (2014). Available at: http://opensource.org/history

Open Source Initiative, *Frequently Asked Questions*, (2014). Available at: http://opensource.org/faq#free-software

Open Source Initiative, *Basics of Open Source*, (2014). Available at: http://opensource.org/faq#commercial

Open Source Initiative, *The BSD 2-Clause License*, (2014). Available at: http://opensource.org/licenses/bsd-license.php

Open Source Initiative, *The BSD 3-Clause License*, (2014). Available at: http://opensource.org/licenses/BSD-3-Clause

A. T. Pham, M. B. Weinstein, J. L. Ryerson, *Easy as ABC: Categorizing Open Source Lincenses*, (2010). Available at: http://www.ipo.org/wp-content/uploads/2013/03/EasyasABC.pdf

Red Hat Inc., *What is Open Source?*, (2014). Available at: http://opensource.com/resources/what-open-source

J. Richards, *The "Abstraction, Filtration, Comparison" Test*, (2002). Available at: http://www.ladas.com/Patents/Computer/SoftwareAndCopyright/Softwa06.html

P. Schmitz, *The European Public Licence (EUPL)*, (2013), International Free and Open Source Software Law Review, vol. 5 issue 2.

P. Schmitz, *EUPL or GPLv3? A Comparison Table of the Main Characteristics and Differences*, (2009). Available at: https://joinup.ec.europa.eu/community/eupl/news/eupl-or-gplv3-comparison-table-main-characteristics-and-differences

N.Shemtov, *The Legal Regulation of Decompilation of Computer Programs: Excessive, Unjustified and in Need of Reform*, (2012). Available at: https://qmro.qmul.ac.uk/jspui/bitstream/123456789/3132/1/SHEMTOVTheLegal2012.pdf

A. Sinclair, *Licence Profile: BSD*, (2010). Available at: http://www.ifosslr.org/ifosslr/article/view/28/62

D. Touretzky, *Source vs. Object Code: A False Dichotomy*, (2000). Available at: https://www.cs.cmu.edu/~dst/DeCSS/object-code.txt

W. H. Venema, *Open Source Software*, National Law Journal, (2008, Oct 20).

M. Välimäki, *GNU General Public License and the Ditribution of Derivative Works*, 2005 (1), The Journal of Information, Law and Technology (JILT).

M. Webbink, *Packaging Open Source*, (2010). Available at: http://www.groklaw.net/article.php?story=20100204170037353

R. Wilson, *The Modified BSD License – An Overview*, (2012). Available at: http://oss-watch.ac.uk/resources/modbsd

Working Group on Libre Software, *Free Software / Open Source : Information Society Opportunities for Europe?*, (2000). Available at: http://eu.conecta.it/paper.pdf

LIST OF CASE LAW

Case Litchfield v. *Spielberg*, 736 F. 2d 1352, 9th Cir. (1984).
Case *Computer Associates International v. Altai*, (1992), 982 F.2d 693, 2d Cir.
Case *Ibcos Computers v. Barclays Mercantile Highland Finance*, (1994), FSR 275.
Case *United States v. Manzer*, 69 F.3d 222, 8th Cir., (1995).
Case 3571, *Sonera Systems Oy v. VF Partner Oy*, (1999), R 99/661.

Case *Nova Productions Limited v. Mazooma Games Limited & Others*, (2007), EWCA Civ 219.

Case C-406/10, *SAS Institute Inc. v World Programming Ltd.*, (2012), (not yet published).