ISSN 2003-2382

Inclusive Rights of Copyleft

A study on the scope of the free and open source software licence in a European context

By Saar Hoek

ABSTRACT

The current legal climate does not yet provide sufficient clarity on the workings, limits and rights conferred through licences granted on Free Open Source Software (FOSS). Generally, copyright is accepted as applying to FOSS and therefore the granted licence; by contrast, whether patents on the computer implemented invention (CII) encompassed in the software are implicitly licensed as well, is less clear. In terms of copyright, this article will examine the lack of clarity and unison concerning certain definitions in the most commonly used licenses, including the scope of the concepts of "distribution" and "derivative" work with GNU's General Public License (GPL) as a guideline. Patent inclusion within the grant will be set out against the nature of patent protection within the field; as well as complications as concerned implied license grants or the lack thereof.

1. COPYLEFT: THE ORIGIN STORY

1.1. Protecting Software

1.1.1. Copyright: Software as a Literary Work

The classic method of intellectual property protection for software is copyright. The choice of copyright originates in the 1980s. Initially, the World Intellectual Property Organization (WIPO) intended to create a *sui generis* protection for software, but this project was abandoned in 1985.[1] While software is mostly functional in nature, patents could not be applied to computer programs at the time in both the United States (US) and Europe, as their respective laws forbade it.[2] Furthermore, the procedure to acquire a patent is lengthy and quite complicated and a more accessible protection was needed. Copyright was the most suitable solution.

In the European Union (EU), this was codified in 1991 in the Software Directive '91[3], and adjusted in the 2009 Software Directive[4]. This text mandated that the European countries would henceforth protect computer programs (in this case meaning the actual code) as 'literary works' in line with the Berne Convention.[5] There are some issues with the choice of copyright for software protection, as software differs greatly from the subject matter that copyright was drafted to protect.

The first issue is with the protection term. A piece of code will automatically be protected for the life of the author plus seventy years.[6] Code is functional and can be paramount to progression in software development. To protect such a work with a monopoly that can last for over a century, in a field where a year is a long time in terms of developments, seems excessive.

Second, software developers write code to serve a purpose. A computer program is a process; it performs a task. Neither the performance nor the task are protected by copyright, just the source code as it is written, as if it were a novel rather than a kind of equipment. This poses real problems in the protection of a work. If an idea for an app is stolen and the exact same app is built using different code, there is nothing the author can do.

In addition, there is an issue of national sovereignty. Although the Software Directive does set out the general scope of protection in the EU, there is no regulation governing copyright. In addition, most licences used on FOSS originate in the US. These facts and the resulting interpretations make copyright for software complicated and impractical, considering its international nature.

Lastly, it is profoundly challenging to ascertain when something is copied code. How many changes must be made for a code to be considered independent, when the change of a single symbol can make a huge difference in the functionality of the software? Certainty regarding permissible actions is crucial when utilising free and open source code, as it will be incorporated into a final product and could pose substantial problems in terms of infringement if handled incorrectly. When using software distributed under a free and open source licence, the question remains to what extent something has to be changed to no longer be seen as a copy or unauthorised utilisation of said software. This issue will be the focus of this article, focusing on copyright and more specifically on what constitutes a derivative work and what is meant by distribution under a FOSS licence.

1.1.2. CIIs and Patent Protection

As mentioned, software has historically been excluded from patentability in many legislations. Under Article 52(2)(c) European Patent Convention (EPC), this is still largely the case, mandating that programs for computers, as such, are is not patentable. However, neither is the

functionality of a program protec-table by copyright. This leaves a gap where protection is needed for the solution that a computer program can provide to a technical problem, without protecting the program as such. The European Patent Office (EPO) refers to this possibility, where the invention is formulated not as a computer program but as a solution whose performance is dependent on a computer (program), as Computer-Implemented Invention (CII).[7] Examples of this may include Graphical User Interface (UI)[8] Inventions, Data Transmission Inventions and Cloud Computing Technology Inventions.[9] The patentability of CIIs has been possible since the *Vicom* case, where the EPO Board of Appeal first decided that although a computer program as a mathematical method cannot be protected, this does not preclude the patentability of a technical process which is carried out under the control of a program[10]. As will be discussed next, this results in an oxymoron.

CIIs are not as clear as one might have hoped. The concept refers to an invention that can be implemented through software, hardware, or both. This creates a somewhat paradoxical loop within the law, because although patents on software are unlawful, it is possible to get a patent on a CII that is implemented solely in software. However, if this was not allowed, CIIs which were implemented in hardware but which *could* possibly be embodied in software would also unavoidably be excluded.[11] Whether this is desirable is a matter unto itself, but it was certainly not the intention behind the law. In a report for the European Commission, the authors even went so far as to say that:

> *'In sum, the term CII is flawed at an ontological level. This may be a confusing conclusion, but it is helpful to prevent even more confusion.'*[12]

A patent grant thus does not result in a cumulative protection with copyright, but instead covers other subject-matter, upon which disparate acts will infringe, even though both subject-matters may be exclusively encompassed by the same software. In the words of the European Commission:

> *'A patent protects an invention as delimited by the patent claims which determine the extent of the protection conferred. Thus, the holder of a patent for a computer implemented invention has the right to prevent third parties from using any software which implements his invention (as defined by the patent claims). This principle holds even though various ways might be found to achieve this using programs whose source or object code is different from each other and which might be protected in parallel by independent copyrights which would not mutually infringe each other.'*[13]

1 Annette Kur, Thomas Dreier, European Intellectual Property Law, (Edward Elgar Publishing Ltd. 2013), 250.

2 Art. 52(2)(c) and 52(3) Convention on the Grant of European Patents of 5 October 1973 (hereinafter EPC).

3 Council Directive 91/250/EEC of 14 May 1991 on the legal protection of computer programs (hereinafter Software Directive '91).

4 Council Directive 2009/24/EC of 23 April 2009 on the legal protection of computer programs (hereinafter Software Directive).

5 Berne Convention for the Protection of Literary and Artistic Works of 9 September 1886 (hereinafter Berne Convention).

6 Council Directive 93/98/EEC of 29 October 1993 harmonising the term of protection of copyright and certain related rights (hereinafter Copyright Duration Directive), Art. 1.

7 EPO Guidelines for Examination, 'Index for Computer-Implemented Inventions' (EPO, 6 March 2017) https://www.epo.org/law-practice/legal-texts/guidelines/cii-index.html accessed 6 December 2020.

8 The user interface is the component of the operating system that enables user interaction. It is the manner in which the software is shown to the user. This might be expressed in the graphical icons for an app or the manner in which the mouse pointer moves on your laptop.

9 European Commission, The Trends and Current Practices in the Area of Patentability of Computer Implemented Inventions within the E.U. and the U.S. (European Union 2016), 10.

10 T 208/84 (computer-related invention/VICOM) of 15.71986 , EP:BA:1986:T020884.19860715, Reasons for the Decision, p. 6.

11 European Commission, Study of the Effects of Allowing Patent Claims for Computer-Implemented Inventions (European Union 2008), 6.

12 Ibid. 7.

13 Proposal for a Directive of the European Parliament and of the Council on the patentability of computer-implemented inventions, rec. 22–23 (as cited in Kur (n 1) 138).

It should be noted that the above quote is taken from a proposed directive on CII, which was rejected. National law is therefore not harmonised with what is stated therein. In fact, to make matters worse, each Member State's national law governs the post-grant life of a patent there is no Unitary Patent (yet). Even if the requirements for patentability are substantially similar, interpretations and principles conceived through a body of national case law in infringement and invalidity cases, for example, have resulted in national approaches that may not be entirely consistent with one another or with the Position of the.[14] This can be exemplified by the principle of technical character. Right now, codification of this exists only in form of the phrase 'all fields of technology' in Article 52(1) EPC, which sets out the substantive criteria for patentability. However, what exactly this means is unclear, as the EPO has used miscellaneous explanations and various terms, including technical effect, technical contribution and further technical effect.[15] We can identify the resulting uncertainty and divergence in the fact that the principle is diligently used in German law, where it originates, whereas a United Kingdom (UK) court has dismissed the argument on technical character as 'something of a counsel of desperation'.[16]

Though Article 52(2)(c) EPC excludes software from patentability, this exclusion is to be read narrowly. The EPO has explained it as such:

> *'A computer program product is not excluded from patentability under Article 52(2) and (3) EPC if, when it is run on a computer, it produces a further technical effect which goes beyond the "normal" physical interactions between program (software) and computer (hardware).'*[17]

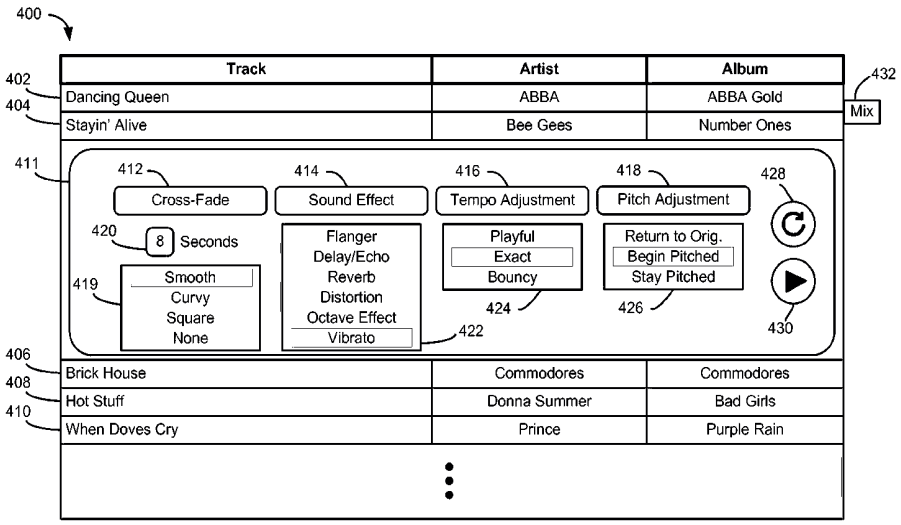
As already mentioned, a proposed directive on the protection of CIIs was rejected. Presently, patent protection is still based solely on the EPC, as well as national laws, revised for compliance with the EPC in light of EPO practice.[18] The lack of harmonised EU law on the patentability of computer programrelated innovation has contributed to the current situation in which there is no clear delineation between a computer program (as such) which may not be protected by a patent and a patentable CII.[19] To avoid the exclusion in in Art. 52(2) EPC and corresponding national provisions, patent applications are often worded in a cryptic manner.

This, in turn, results in a few issues. First, because there is no true classification for patents such as these, it is difficult – even nearly impossible – to find them effectively and thereby ascertain the state of the art.[21] This means that granted patents relevant as prior art might often be overlooked, even in case of diligent research, and the amount of patents might therefore increase unjustly, thereby lowering the quality of patents. Another result of inef-

Figure 1.
A patent application filed by Spotify for which a patent was granted. The abstract explains the process which is enacted by a computer. This invention is entirely embodied in software.[20]

(54) **Crowd-sourcing of automatic music remix rules**

(57) Systems and methods for mixing music are disclosed. Audio mix information is received from a plurality of users. Mix rules are determined from the audio mix information from the plurality of users, wherein the mix rules include a first mix rule associated with a first audio item. The first mix rule relates to an overlap of the first audio item with another audio item. The first mix rule is made available to one or more clients. In some implementations, making the first mix rule available to the one or more clients includes transmitting, to the first client, information enabling the first client to playback a transition (e.g. crosss-fading) and a musical effect (e.g. reverberation) between the first audio item and a second audio item in accordance with the first mix rule.



fective searching is that it is difficult to ascertain whether a certain type of conduct infringes upon a certain patent, if said patent is hard to find. Lastly, because patents protect and therefore formulate an idea rather than an expression, it is not required that source code is disclosed. This is curious, as Article 100(b) EPC clearly states that any patent application must disclose the invention in such a manner that a person skilled in the art might carry it out. The solution in such a patent is often entirely encrypted in computer code, and its absence in a patent specification would make it a burdensome task for the person skilled in the art to carry out the invention. Additionally, one might argue that due to this type of reasoning, the patents granted for CIIs might be unduly broad, and thereby not fulfilling the requirements of inventive step in Article 56 EPC.[22] In fact, in the case of software-implemented inventions, it is often not a specific solution that has been granted a patent but rather a specific problem, because the expression – the source code, the computer program itself, which enacts the solution – is excluded from patentability.[23]

1.2. Conception of Copyleft

1.2.1. From Sharing to Selling

> *'When I started working at the MIT Artificial Intelligence Lab in 1971, I became part of a software-sharing community that had existed for many years. Sharing of software was not limited to our particular community; it is as old as computers, just as sharing of recipes is as old as cooking.'*[24]
>
> – Richard Stallman

Historically, source code has been provided along with whatever program it represented. Indeed, back at the beginning of software development, the attitude towards its identity and accessibility was built upon long traditions of collaboration and openness.[25] These attitudes existed because software was seen as a means to an end – the end being hardware and software being a mere necessity to make it function, and not possessing any independent value. One contributing reason was that the system of updating and tweaking technology was very different from the one-click-culture that we enjoy today. As a user of "primitive" technology (pre-1980s), it was important that the source code to software was freely accessible as it might be necessary to update and modify it yourself in order to support new hardware or add features.[26] Personal computers had not entered the landscape yet and computing was still intrinsically a thing of science and education. As this went on for years, it created a norm that was hard to depart from, and it can be argued that the principle of free access is still present among programmers.

However, a shift occurred in the 1980s. Software started being pursued as a business, which meant that the willingness to share proprietary source code decreased significantly. For the people working in software, this was a deeply frustrating experience. Not only did the amount of easily accessible work material decrease; they were also suddenly subject to non-disclosure agreements and lawsuits where the culture used to be open collaboration. One of these frustrated programmers was Robert Stallman. In his own words:

> *'This meant that the first step in using a computer was to promise not to help your neighbour. A cooperating community was forbidden. The rule made by the owners of proprietary software was, "If you share with your neighbour, you are a pirate. If you want any changes, beg us to make them."* '[27]

1.2.2. Copyleft and Licence Types

To combat increasing proprietary approaches and keep software free (as in 'free speech', not as in free from cost), Stallman founded the Free Software Foundation (FSF), under which he started developing a new operating system called "GNU's Not Unix!" (GNU) and conceived the concept of copyleft. The idea is based on copyright, but takes it in reverse. Instead of using it as a means to privatise and monopolise a work, it ensures that the work remains in the public domain. Under copyleft, one is free to use, distribute, modify and copy the program – but one is *not* allowed to add subsequent restrictions. What is free must remain so; if a copylefted program is incorporated into another program, the source code must be included, including any independently made modifications or additions.[28]

[14] European Commission (n 11) 11.
[15] Ibid. 15.
[16] CFPH LLC [2005] EWHC 1589 Pat. [2006], R.P.C. 5.
[17] T-1173/97 (Computer program product/IBM) of 1.71998, , EP:BA:1998:T117397.19980701, Reasons for the Decision, p. 2.3-2.4.
[18] Anna Haapanen, Free and Open Source Software Licensing and the Mystery of Licensor's Patents (IPR University Centre, 2017), 73.
[19] Kur (n 1) 139–144.
[20] Spotify AB, 'Crowd-sourcing of automatic music remix rules', EP2808870A1, granted 16 March 2016.
[21] Directorate-General for Internal Policies of the European Parliament, Legal Aspects of Open Source Software (Policy Department C: Citizen's Rights and Constitutional Affairs Workshop, 2013), 45.
[22] T 939/92 (Triazoles) of 12.91995, , EP:BA:1995:T093992.19950912, Reasons for the Decision, p. 2.4.2.
[23] European Parliament (n 22) 45.
[24] Robert Stallman, Free Software, Free Society (FSF 2002), 23.
[25] Ibid. 1–3.
[26] Ibid 1.
[27] Stallman (n 24) 18.
[28] GNU's Not Unix, 'Frequently Asked Questions about the GNU licenses' (GNU Operating Systems, updated 9 February 2019) https://www.gnu.org/licenses/gpl-faq.html accessed 9 March 2020.

Therein lies the distinction between free software and copyleft. A licensed program can satisfy the requirements for the former but not for the latter, e.g., when a piece of code can be freely included in a proprietary program *without* the condition that its source code must be included. Such FOSS licences are generally referred to as permissive licences. The difference boils down to that the user further along the distribution chain, who obtained the initial open source software only through its incorporation in other software, is still subject to any original copyleft licence. Permissive licences entail no such restrictions. The copyleft licence is inherently connected to the software and travels with it, while permissive licences can be replaced.

Stallman drafted his copyleft ideas in the General Public License (GPL), which is widely used to this day. Other popular modules include the MIT licence and the Apache licence, both permissive licences.[29] The Open Source Initiative (OSI), which is an organisation that approves FOSS licences, is also noteworthy. They opt not to use the term 'free software' but rather 'open source', because – though they have similar goals and history as the FSF – their foundation is pragmatic and business-oriented, rather than ethical in nature, and they have felt that the term 'free' has too many moral connotations.[30] Here, both terms will be used synonymously.

1.2.3. Compatibility

With any licence, a large amount of freedom is granted to its drafter. This has resulted in a multitude of free software licences, not all of which are compatible with each other.[31] However, unlike in most fields where licences are used, the field of software is highly collaborative and strangers across the globe can – and will – easily make use of its subject-matter. The chain of distribution and adaptation is almost impossible to track, which has made it difficult to keep track of how the licences work in practice, how they affect each other, how an American-written copy-pasted licence applies in an EU state and so forth. A study done in 2013 suggested that many GitHub[32] users did not license their source code at all.[33] The situation has grown so complex that it is very hard to comprehend or get an overview of. For the sake of clarity, this article will adopt the GPL as connecting theme throughout. This means the focus will be on copyleft licences and the licences compatible with GPL. The reason it that, a copyleft licence poses strict obligations on its licensee, such as the publishing of source code, which a permissive licence does not. This means that disputes and legal uncertainties regarding its scope are more consequential. The GPL is the obvious choice, as it is widely used, widely discussed, has very strict obligations and is the original copyleft licence.

1.3. The European Union

1.3.1. Legal Basis

It may seem counterintuitive that a contractual solution originating in US Copyright Law can function in European countries (and it does not completely, see section 1.3.3.), but because it is a contract it does not supersede copyright law. Rather, it is an overlying agreement, which grants the user and proprietor certain rights and obligations. Any infringement will be settled under the relevant national law. In fact, national legislation varies quite a bit between the EU member states, as illustrated in Figure 2 below. Note that at the time of the investigation – 2016 – there were still many countries in which no case law existed on FOSS or alternative licences (such as the Creative Commons licence). In the countries that did have some reported case law, this usually encompassed only one or two cases.[34]

In the absence of extensive case law, an analysis of the terms used in most copyleft licences will be helpful to get a picture of what happens when FOSS is licensed. In Figure 2, under the question 4, it is noted that eleven EU states have jurisdiction-specific standard licences for FOSS. In all these cases, the licence in question is the so-called European Union Public License (EUPL) for FOSS.[36]

[29] Ayala Goldstein, 'Top 10 Open Source Licenses in 2018: Trends and Predictions' (White Source Software, 13 December 2018) https://resources.whitesourcesoftware.com/blog-whitesource/top-open-source-licenses-trends-and-predictions accessed 10 March 2020.

[30] Michael Tierman, 'History of the OSI' (OSI, 19 September 2006) https://opensource.org/history accessed 9 March 2020.

[31] GNU operating system, 'Various Licenses and Comments about Them' (GNU Operating System, 13 March 2017) https://www.gnu.org/licenses/license-list.en.html accessed 9 March 2020.

[32] The most commonly used website where programmers up- and download freely accessible source code.

[33] Aaron Williamson, Licensing of Software on GitHub: A Quantitative Analysis, Linux Collaboration Summit, 2013.

[34] Axel Metzger, Free and Open Source Software (FOSS) and Other Alternative Licensing Models (Springer, 2016), 7–12.

[35] Metzger (n 35) 6.

[36] Metzger (n 35) 12.

[37] Patrice-Emmanuel Schmitz, 'The European Public Licence (EUPL)' [2013] 5(2) International Free and Open Source Law Review, 121.

[38] Software as a service means that a program is not downloaded or bought on an external disc or drive, but rather functions through a cloud computing system and is accessed via the internet. The Citrix web environment is an example.

[39] As most European courts operate on a civil law system rather than common law, a contract is a little less free. For example, a general exception to liability is not accepted in most European courts.

[40] Schmitz (n 37) 122.

[41] EUPLv1.1, European Union Public Licence version 1.1 (European Commission, January 2007) https://joinup.ec.europa.eu/sites/default/files/custom-page/attachment/eupl1.1.-licence-en_0.pdf; European Union Public Licence version 1.2 (European Commission, May 2017) https://joinup.ec.europa.eu/sites/default/files/custom-page/attachment/eupl_v1.2_en.pdf accessed 9 March 2020.

[42] Schmitz (n 37) 122.

[43] Javier Casares, 'EUPL: European Union Public Licence' (EUPL, updated 27 June 2017) www.eupl.eu accessed 9 March 2020.

Figure 2.
Additional information on national provisions and case law regarding FOSS and alternative licensing in EU Member States and UK.[35]

	Yes	No	Yes Countries	No Countries
1. Special provisions on license contracts?	12	5	Belgium, Croatia, Czech Republic, France, Germany, Greece, Hungary, Italy, Poland, Portugal, Romania, Spain	Cyprus, Denmark, Finland, Netherlands, UK
2. Special provisions on FOSS or other alternative licenses?	5	12	Czech Republic, France, Germany, Italy, Portugal	Belgium, Croatia, Cyprus, Denmark, Finland, Greece, Hungary, Netherlands, Poland, Portugal, Romania, UK
3. Case law on FOSS or other alternative licenses?	6	11	Belgium, France, Germany, Italy, Netherlands, Spain	Croatia, Czech Republic, Cyprus, Denmark, Finland, Greece, Hungary, Poland, Portugal, Romania, UK
4. Jurisdiction-specific standard licenses for FOSS or other jurisdiction-specific licensing schemes?	11	6	Belgium, Czech Republic, Finland, France, Germany, Italy, Netherlands, Portugal, Romania, Spain, UK	Croatia, Cyprus, Denmark, Greece, Hungary, Poland

1.3.2. European Union Public Licence

In the early 2000s, the European Commission started to review the advantages of adopting a licence for open source software. This occurred mainly in the context of programs meant to improve interoperability (within EU institutions and the public sector) and in relation to the development of the information society.[37] The Commission first set out the eight conditions the chosen licence would have to encompass, namely:

1. Grant *all* FOSS freedoms;
2. Ensure protection from exclusive software appropriations (i.e., be a copyleft licence);
3. Have working value in *all* official EU languages (so as to avoid the need for sworn translators);
4. Conform with EU copyright law and terminology;
5. Include the 'communications to the public' right, including web distribution and software as a service[38];
6. Clarify applicable law and competent court;
7. Have an approach to warranties and liabilities that conforms with case law[39];
8. Be comprehensive and pragmatic, avoid complexity and excessive length.[40]

Research found that no existing licence complied with four of these requirements (3, 4, 6 and 7). Already, this reveals something about the GPL in a European context, as it was one of the licences considered, and therefore apparently did not comply with the requirements that the Commission deemed necessary – most importantly requirement 4.

It was decided that the best option was to create a new copyleft licence, which came to be the EUPL[41]. Version 1.1 was released in January of 2009 and accepted – in all its 22 languages – by the OSI in March of the same year.[42] It has grown popular primarily within governmental institutions and public service organisations, as many countries in the EU require that the local language be used at such institutions.

The licence is compatible with the GPL. However, keep in mind that this means that if the two licences are combined, the combined product has to be licensed under the GPL, as this is one of the main requirements of the GPL.[43] The purpose of the EUPL was never to compete with existing licences, but rather to facilitate the use of FOSS in European public governance. It is therefore not useful to do a side-by-side comparison of the EUPL and the GPL. However, because one of the main purposes of the EUPL was to make a licence which would be compatible with EU law, terminology and case law, it will be useful to use it as guidance to see how best to interpret the GPL (and other copyleft licences), for example as regards liability.

1.3.3. Liability and Warranty

It is worth mentioning that most FOSS licences contain an absolute disclaimer on warranty and liability. An example is seen in the following, taken from the MIT licence, which is currently the most popular licence on GitHub:[44]

> *'The software is provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software.'*[45]

This is a typical example of a clause which is obviously derived from common law. Such a clause will not hold up in most courts of the EU, as shown in Figure 3. Even for the Member States which are shown in the 'yes' column, validity will not be absolute. For Belgium and Finland, liability and warranty in case of gross negligence or wilful acts cannot be excluded. For the Netherlands, Croatia and Finland, such claims may be void in cases concerning consumers.[46] Most Member States even have specific mandatory provisions prohibiting such claims.[47] In court, such a provision will then be declared void, meaning the author could be liable for damages. However, assigning liability might be difficult in many cases, due to the interoperability of software; a failure might be due to connected software or hardware.[48]

In the EUPL v1.2, the liability clause is rephrased as follows:

> *'Except in the cases of wilful misconduct or damages directly caused to natural persons, the Licensor will in no event be liable for any direct or indirect, material or moral, damages of any kind, arising out of the Licence or of the use of the Work, including without limitation, damages for loss of goodwill, work stoppage, computer failure or malfunction, loss of data or any commercial damage, even if the Licensor has been advised of the possibility of such damage. However, the Licensor will be liable under statutory product liability laws as far such laws apply to the Work.'*

Such a clause fits the civil law of most EU Member States much better, and it is likely that most liability disclaimers will be interpreted in this way, even if they expressly exclude all liability. Therefore, authors and distributors need to be cautious when locating or conducting business in one of these territories, because the risk for liability might be larger than assumed. Indeed, especially in most cases of wilful misconduct or gross negligence, liability cannot be avoided.

2. RELATIONSHIP COPYRIGHT AND FOSS

2.1. Rights Conferred

The OSI has established some general rules as to which rights have to be included in a copyleft licence. Although there are also many non-OSI certified licences, the rules are still generally adhered to. Even if this were not the case, the most commonly used licences, such as the GPL, examined here, have been approved. Thus, for the purposes of this article, these rights provide excellent guidance.[50]

The rights required by the Open Source Definition (OSD) are: to use, reproduce, modify, communicate and re-distribute the work.[51] These refer, of course, to the rights granted to the author of the work copyrighted in the first place, under the Berne Convention.[52] For software in the EU, this has been established in Article 4 of the Software Directive, with which national legislation will have been harmonised. Use, reproduction, modification

Figure 3.
Validity of exclusion of any liability and warranty claims in some EU Member States.[49]

	Yes	No
1. Is a disclaimer excluding any warranty and liability (such as often occurs in FOSS) valid under the contract law principles of national jurisdiction?	Belgium, Croatia, Denmark, Finland, Netherlands	Czech Republic, France, Germany, Greece, Italy, Poland, Portugal, Romania, Spain
2. Is it of relevance that the license grant of FOSS and other alternative licensing schemes is not bound to any monetary consideration?	Denmark, Netherlands, Spain	Belgium, Croatia, Czech Republic, Germany, Hungary, Italy, Romania

and communication will not be discussed herein. Use and modification are quite straightforward and do not require elaboration within this context. Reproduction and communication are interesting in a software context, but issues such as piracy and the communication of paid content on a free site, for instance, are not particular to or larger in copyleft licences as compared with other copyright issues and thus will fall outside the scope of this article.

2.2. Distribution

2.2.1. Accessibility Requirement

Distribution is different from the other rights because it is the only condition within a copyleft licence that imposes an obligation on the user. Namely that what is free must remain free. It is not the particularity of distribution in and of itself that is the key component. Rather, it has to do with the fact that any software licensed under the GPL must remain licensed under the GPL.[53] Therefore, if a company uses any software licensed in this way, said software must be published within their own product. This is the case even in a compiled binary program made up of many files, if the vast majority are licensed under a permissive licence and only one is under a copyleft licence. Permissive licences such as the BSD licence[54] allow sub-licensing and defer to the GPL when used in combination. This practice is referred to as deep-licensing.

The requirement which forces free accessibility is only complicated *when* distributing, due to companies wanting to sell – and therefore to distribute – finished products that contain some type of copyleft-licensed software. In case of modification or reproduction, no such obligation arises. Notice can be provided in a multitude of ways. For apps and other 'clean software', it is common that there is a section called 'Third Party Software'[55] or something similar. For hardware running GPL software, this is a bit more complex – but one can implement a notice in the UI or provide notice in the accompanying documentation.[56]

A company should make certain that these notices are diligently provided, because if FOSS licensed under the GPL is used within proprietary soft- or hardware and a licensee has failed to license the combined product under the GPL and not provided due credit, that is infringement of the licence, which means liability for copyright infringement.

2.2.2. Offer and Acceptance

In the EU member states, a licence agreement generally must adhere to the conditions of offer and acceptance.[57] The publishing of FOSS with a copyleft licence can certainly be accepted as constituting an offer. In the first paragraph of the GPLv2, this offer is subject to the condition that the licensee

> *'conspicuously and appropriately publish on each copy an appropriate copyright notice' as well as 'keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program'.*

Moreover, it is stated in the fifth paragraph that the act of modifying or running the program constitutes acceptance to these terms and conditions. What this means for the licensee is that if they distributes certain parts of FOSS that had been licensed under the GPLv2, they have accepted the licence terms. This will probably not be disputed by a licensee, because without acceptance there would not be a licence in the first place. If they have not distributed the work under the GPLv2, the terms of the offer are not adhered to and there will have been no licence agreement. In fact, the GPL has a clear termination notice:

> *'You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License.'*

In these cases, there will thus have been no right to use the software at all, which means that all uses of the software falling under any of the exclusive acts in Article 4 of the Software Directive – or in some cases Articles 2–4 of the InfoSoc Directive – even including temporary reproduction, such as loading, would constitute copyright infringement. In the case of the GPLv3, the licence may be re-instated if the error is rectified. However, failure to do so within 30 days of notice means that the licence will be permanently terminated.[58]

Distribution is particular, too, in the sense that it can be exhausted after a first sale or transfer of ownership pursuant to Article 4(2) of the Software Directive (see section 2.5 below).

---

44 Ben Balter, 'Open Source License Usage on GitHub.com' (GitHub, 9 March 2015) https://github.blog/2015-03-09-open-source-license-usage-on-github-com/ accessed 9 March 2020.

45 MIT licence, para. 3.

46 Metzger (n 35) 24.

47 See for example Czech Republic, Sec. 2898; France, Art. 1386-1 CC; Germany, Sec. 309 no. 8 lit. b and no. 7 CC.

48 Victoria Ho 'EU Software Liability Law Could Divide Open Source' (CNET, 11 June 2009) https://www.cnet.com/news/eu-software-liability-law-could-divide-open-source/ accessed 9 March 2020.

49 Metzger (n 35) 24.

50 OSI, 'Licenses by Name' (OSI, updated 2 May 2019) https://opensource.org/licenses/alphabetical accessed 9 March 2020.

51 Schmitz (n 37) 124.

52 Berne Convention Art. 6–13 pertaining to literary works.

53 GPLv2, section 0; GPLv3, section 2.

54 Berkeley Software Distribution License modified version (UC Berkeley, July 1999) https://opensource.org/licenses/BSD-3-Clause accessed 9 March 2020.

55 The Spotify app can serve as an example: you can find a list of their utilised copyleft-licensed software with the following steps: Your Library Settings About Third-party software.

56 Fredrik Öhrström, Software Patents and Free Software, (Stockholms Universitet Lecture, Stockholm, November 2018).

57 Haapanen (n 18) 89.

58 GPLv3, Art. 8(3).

2.3. Infringement Repercussions

The stakes in case of infringement are higher than one might think. It is natural to assume the risk is low because FOSS can often be acquired for free – in the legal sense of the word – so the damage would be nil in terms of actual financial loss on the author's part. However, in no EU country does this prevent the author from filing for damages if the licensee does not comply with the conditions of the licence. The interpretation of what those damages might be differs throughout the EU depending on national approaches. In some jurisdictions, such as Hungary, relevant loss might need to be proven, which could be problematic. Other countries, such as Denmark and Germany, might award damages based on fees for similar licences or for comparable software, in which case the amounts could be substantial.[59] When there are interests in the US, which is very possible considering the global nature of software, the damages could prove to be greater. An example can be seen in the 2017 US case in which CoKinetic Systems Corporation filed suit against Panasonic Avionics Corporation. Both companies are *global* players in the in-flight entertainment market. The claim was that Panasonic had wilfully violated the GPLv2 requirements by refusing to provide source code. Panasonic – holding a dominant market share of about 70% – was accused of attempting to monopolise the market. The damages sought exceeded $100 million.[60]

Interestingly, this case was settled in 2018. It is near impossible to find cases involving FOSS licence infringement that have not been settled. This is best explained by looking at the core of the conflict, which concerns not the monetary repercussions, but the assets. Recall that FOSS is essential in technological development and how widespread it is. Open source is used in creating almost every computer program. Institutions such as the European Commission, conglomerates like Microsoft, engineers, programmers, hardware manufacturers, research centres at universities, leaders in AI – *almost everyone uses open source*.[61] Reasons to use open source are myriad: it saves a lot of cost in development and due to its accessibility has been checked, bug-fixed and improved upon by more experts than a sole company could ever hope to afford. Reports have been released from all sectors explaining the need for the use of open source.[62]

Now imagine the effort, research, cost and time it takes to develop advanced technology. This is illustrated in the following quote Bill Joy, co-founder of Sun Microsystems, Inc., which has been bought by Oracle:

> *'We spent over a billion dollars a year in research. I can't just throw it all on the street.'*

Herein lies the crux of the matter. The technology does not just *have* worth, it is the worth. If a company is found to be in breach of the GPL or another copyleft licence, resulting in a lawsuit, the risk is having to give out any separately developed adjacent or encompassing source code – which most often is the source of profit. This risk arises because the GPLv3 speaks of 'covered work' and the GPLv2 and EUPL of 'derivative work'; it is unclear what the scope of these terms are and therefore how much of proprietary code would have to be released.[63] This is a daunting prospect and could undo millions in research and years in development.

There is also a risk that a product may need to be recalled. The Enforcement Directive states that in case of infringe-

[59] Metzger (n 35) 38.

[60] CoKinetic Systems, Corp. v Panasonic Avionics, Corp 1:17-cv-01527 (S.D.N.Y. 2017).

[61] Öhrström (n 56).

[62] See for example European Commission, Report on Open Source Licensing of Software Developed by the European Commission (hereinafter Commission FOSS Report), (European Union 2004), p. 4., and Sören Sonnenburg and others, The Need for Open Source in Machine Learning [2007] 8 Journal of Machine Learning Research, 2449–2453.

[63] GPLv3, 5(c).

[64] Directive 2004/48/EC of the European Parliament and of the Council of 29 April 2004 on the enforcement of intellectual property rights [2004] OJ L 157 (hereinafter Enforcement Directive), Art. 10(1).

[65] Maribel Lopez 'Samsung Explains Note 7 Battery Explosions, and Turns Crisis into Opportunity' (Forbes, 22 January 2017) https://www.forbes.com/sites/maribellopez/2017/01/22/samsung-reveals-cause-of-note-7-issue-turns-crisis-into-opportunity/#28f4f40624f1 accessed 9 March 2020.

[66] Not intended in the sense of someone performing 'digital breaking-and-entering', but as a description of those creating and working with software etc.

[67] Robert Gomulkiewicz 'De-Bugging Open Source Software Licensing' [2003] 64(1) University of Pittsburgh Law Review, 75.

[68] Cristoph Helwig v VMware Global, Inc. Zweigniederlassung Deutschland, Hamburg District Court 310 O 89/15, 8 July 2016.

[69] A kernel is the core of an operating system. It is the computer program that is the most essential to the entire system, exercising complete control. You can see it as the brain of an operating system.

[70] Application Programming Interface. Every website on the internet is stored on a remote server. These are not mystical clouds of information, but actual tangible computers somewhere on the planet. If you type a website's URL into your browser, a request goes out to its computer, the "server". The part of the server that handles such requests and sends responses is the API. It is not the entire remote server, rather the part that your query interacts with.

[71] Ieva Giedrimaite 'VMware GPL case is back in court—will we finally get some clarity on the meaning of "derivative work"?' (IPKat, 28 January 2019) http://ipkitten.blogspot.com/2019/01/vmware-gpl-case-is-back-in-courtwill-we.html accessed 9 March 2020.

[72] Ibid.

ment of intellectual property, the Member States shall implement corrective measures including recall from commerce, definitive removal from commerce or destruction – on top of damages.[64] A company would not want to risk having to recall an entire product line, which might be the case if the software used is embedded into hardware. If we take the example of the Samsung Galaxy Note 7, the mobile device that had a battery which was prone to spontaneously exploding, recall of 2 million devices cost Samsung an estimated $5.3 billion.[65]

Lastly, for the party filing the lawsuit, there can be more business advantages from settling a case with the alleged infringing party, if the plaintiff agrees to provide them with valuable information and/or source code, which would result in a stronger market position for both parties than in case of public disclosure. It is a win-win for the parties, but a loss for open source.

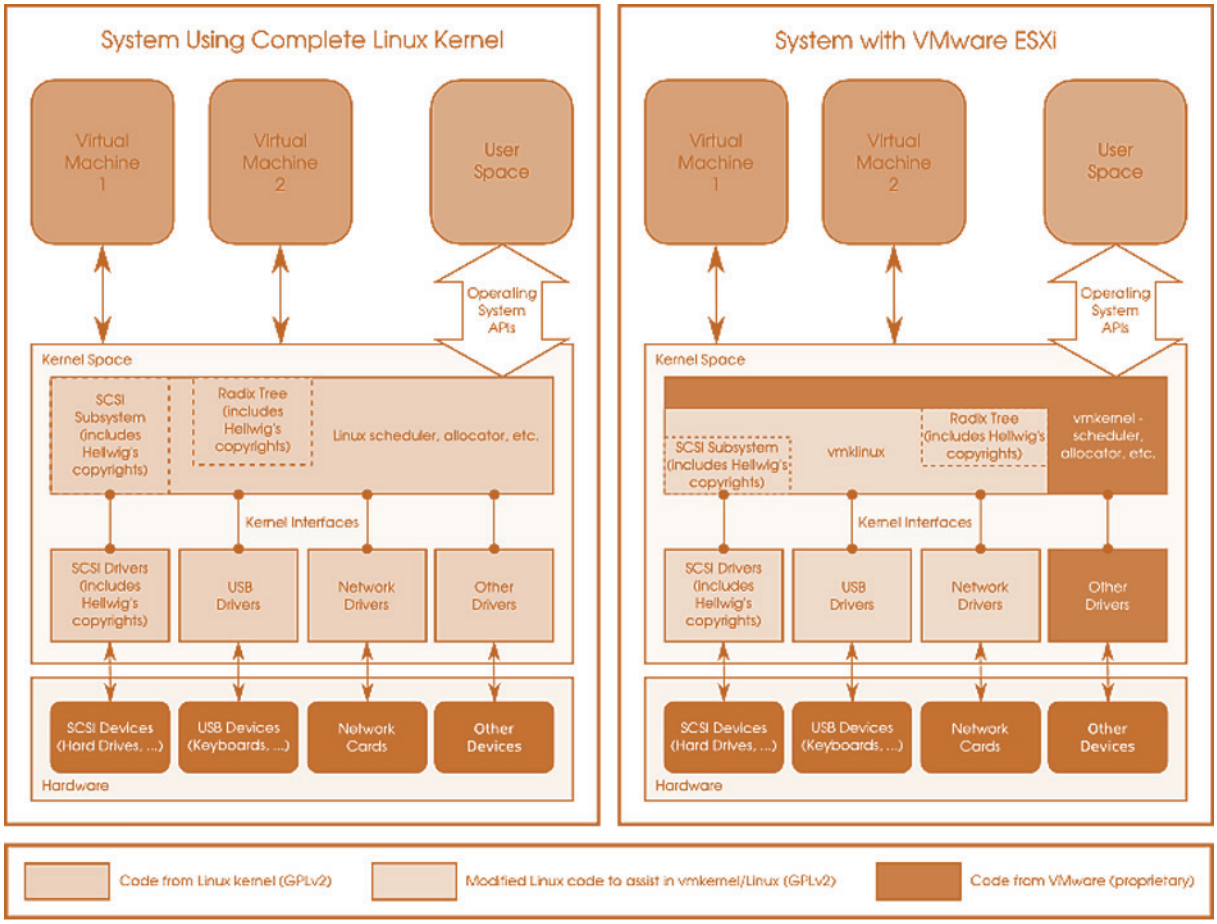
Due to the high risks of litigation, the unclarity of the terms and the benefits of settlement, very few cases have made it through to judicial rulings, meaning that many aspects are yet to be clarified. The situation has resulted in many companies being hesitant to use copylefted software – putting them at a disadvantage, because of the vast amount of resources that thereby become unavailable to them. This, in turn, affects the speed of technology development and the fairness of the playing field. Right now, hackers[66] suffer because they do not know which licence to use, end users suffer because they do not understand the terms of the licences, and companies suffer because they do not understand how open source might affect their intellectual property.[67] As illustrated above, what is meant by 'covered work', 'derivative work' and other similar terms is essential in relation to copyright.

2.4. Derivative Work

2.4.1. VMware v Hellwig

To get an idea of the complexity to be dealt with, let us examine an example: the case of *VMware v Hellwig*.[68] Linux was the proprietor of a kernel[69], licensed under the GPLv2. The opposing party, VMware, was the proprietor of another kernel, the vmkernel, as well as an API[70] called VMK API. Third parties were able to write drivers which would interact with the VMK API. For Linux drivers, an alternative compatibility option was offered through a loadable kernel called vmklinux. These three facets together, vmkernel, VMK API and vmklinux, were all encompassed in the ESXi OS.[71] vmklinux was licensed under the GPLv2, but the ESXi system was available only under a commercial licence.

Figure 4.
A system using the complete Linux kernel compared with a system with ESXi. The vmkernel is connected to the vmklinux and serves as a type of wrapper.[72]



The GPLv2, in term 0, states that:

> *"This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language."*

Further, in term 2(b), it states:

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

The essential takeaway is that any derivative work must be licensed under the GPLv2. What, then, is meant by a derivative work? A work containing another work is usually seen as a derived work, but here two separate definitions have been used: 'that in whole or in part contains or is derived from [...]'. Does the licence mean to widen the term to encompass code and possibly even data or ideas that are otherwise not protected under copyright law? Unfortunately, the case in question has been dropped, in part because VMware promised to remove its allegedly Linux-derived technology from its OS. Thus, a definitive and prompt answer will not be forthcoming, but the case – which spanned over a decade – does illuminate the kinds of issues that arise. If a component of a software is licensed under the GPL, does that 'contaminate' the rest of the software? How much interaction is allowed? After how many modifications can a work be seen as independent?

2.4.2. Legal Definition

Regrettably, the EUPL does not offer much help. Its definition of derivative works is as follows:

> *'The works or software that could be created by the Licensee, based upon the Original Work or modifications thereof. This Licence does not define the extent of modification or dependence on the Original Work required in order to classify a work as a Derivative Work.'*

This comes down to that a work, to qualify as a derived work, must be derived from an original work. This is not a very helpful definition, especially given the fact that copyright in software development is unique, in that it is necessary to utilise already available work. Avoiding this would be like requiring that every car manufacturer reinvent the wheel every time a new model is created. Moreover, although it has been pushed into a characterisation as a literary work, software is still functional. This means that pieces of code work together to achieve a common goal. In the GPL explanatory notes, it is stated that an aggregation of a program is not protected under the GPL. In other words: when a piece of GPL-licensed software is used or modified in a program, in which it interacts with independently owned code, the licence does not apply to the resulting proprietary code. But where should the line be drawn between the pieces of software? The FSF GPL FAQ states that this depends on how 'intimate' two programs are.[73]

Of course, software giants want to make their products as accessible as possible. This means that they want to be able to process signals from already established software. This is especially true for software new to an established market, which can be exemplified by the failure of the Windows Phone OS.[74] The failure had little to do with the soft- or hardware of the phone itself, but rather that the new operating system (OS) was unable to process most apps written for the established Android and iPhone OSs. A user of Windows Phone would thus be unable to participate in the state of the art, resulting in the OS flopping. This issue could easily have been fixed by interoperability. VMware attempted to promote this by making its own software interoperable, but of course the software *need* to work intimately together. That could mean that the entire system would be a derivative work and the source code would have to be released in order to avoid copyright infringement and damage claims.

If we look to the Software Directive, it hints at what could constitute a derivative work in Article 4(1)(b), which states the exclusive right of an author to authorise:

> *'the translation, adaptation, arrangement and any other alteration of a computer program and the reproduction of the results thereof, without prejudice to the rights of the person who alters the program.'*

This is a broad interpretation, as it denotes any kind of alteration. Moreover, it is unclear what translation means in this context, as computer programs are not written in human language, but rather in source code. This could mean that a translation could be a reformulation of a program in another programming language. This would make the definition problematic, because that would mean that the idea or essence of the program is protected rather than the actual code – we have established that this is not protectable by copyright. The article does mention that the rights of the person who alters the program are unaffected, but this is not the case for a copyleft licence, which would impact precisely those rights – in the case of the GPL they are often waived and assigned to the FSF.

Traditionally, a derivative work is subject to two conditions: there must be a pre-existing work that it is based on and a separate original contribution thereto.[75] Like the Software Directive, neither national legislation nor the Berne Convention provide exhaustive lists on what can constitute a derivative work.[76] Rather, the lists provided serve as illustration. The pre-existing work is the copyleft-licensed FOSS that is used, and the separate original contribution is the work that a company may want to distribute. Looking to the teleological context of a law regulating derivative works and alterations, we can conclude that it generally concerns versions of a work that are

directly connected to said work. A translation or cinematographic adaptation of an original literary work is clearly a version of that work, even if an original contribution is made to it and protected independently. Therefore, taking into account the nature of software, it is unlikely that the law meant to include any sort of interconnected but independent software as a derivative work, unless it was based on the precise code itself. Recital 15 of the Software Directive supports such a view:

> *'The unauthorised reproduction, translation, adaptation or transformation of the form of the code in which a copy of a computer program has been made available constitutes an infringement of the exclusive rights of the author. Nevertheless, circumstances may exist when such a reproduction of the code and translation of its form are indispensable to obtain the necessary information to achieve the interoperability of an independently created program with other programs. It has therefore to be considered that, in these limited circumstances only, performance of the acts of reproduction and translation by or on behalf of a person having a right to use a copy of the program is legitimate and compatible with fair practice and must therefore be deemed not to require the authorisation of the rightholder. An objective of this exception is to make it possible to connect all components of a computer system, including those of different manufacturers, so that they can work together. Such an exception to the author's exclusive rights may not be used in a way which prejudices the legitimate interests of the rightholder or which conflicts with a normal exploitation of the program.'*

Clearly, the aim of the Directive is to promote the software industry and provide solid legal protection for those investing in development. Additionally, a reading of the recitals reveals that the aim of the Directive is not to hinder any type of use of software which is necessary for interoperability. Any other interpretation would leave very little room for build-on technology and thereby stifle innovation.

2.4.3. Evaluating a Work

As for many legal issues, the question if a work is derivative should be evaluated on a case-by-case basis. However, the evidence points toward a more narrow interpretation than what was argued in the *VMware v Hellwig* case. To include an entire OS as a derivative work of a kernel because it attempts to be interoperable with said kernel would negate exactly such investments that the Software Directive and other copyright protection for computer programs set out to protect. Of course, if the code for the ESXi system had been substantially similar to the Linux OS, this would have been different. A derivative work in this context could be defined as a work based on an original work, in its entirety or in part, unless (a) the part concerned exists exclusively to aid in interoperability and promote a harmonised software environment; and (b) the connected software has been developed independently.

As this is a difficult evaluation to make, it might prove useful to erect a legal fiction of a person adept at analysing the similarity or intimacy between the original and the allegedly derived work. This would correspond to the 'person skilled in the art' in patent law, the 'average consumer' in trademark law and the 'informed user' in design law.[77] Naturally, copyright differs in nature, as it is an unregistered right.[78] However, due to its functional nature in the case of software, a parallel can be drawn specifically with the inventive step assessment in patent law. Here, it must be emphasised that it is the expression of the code that would be assessed, not the idea behind it or the functionality thereof. This is a distinction that such a person would have to comprehend. In patent law, the person skilled in the art has the twin tasks of preventing trivial inventions from being patented and preserving the patentability of meritorious ones.[79] When applied to a software environment – specifically in the context of copyleft licensing – such a person could have the mirrored twin tasks of preventing copycats while simultaneously safeguarding original contributions, notwithstanding the existing legal framework in regard to copyright protection for computer programs.

[73] Gomulkiewicz (n 67) 91.

[74] Vlad Savov, 'Windows Phone was a Glorious Failure' (The Verge, 10 October 2017) <https://www.theverge.com/2017/10/10/16452162/windows-phone-history-glorious-failure> accessed 9 March 2020.

[75] Ramū nas Birštonas, 'Derivative Works: Some Comparative Remarks from the European Copyright Law' [2013] 5 University of Warmia and Mazury Law Review, 67.

[76] Berne Convention, Art. 12; Ibid. 68.

[77] Naina Khanna and Jasmeet Gulati, 'Knowledge/Skill Standards of a "Person Skilled in Art": A Concern Less Visited' [2018] 17 John Marshall Intellectual Property Law Review, 590.

[78] Berne Convention Art. 5(2).

[79] Ibid. 591; EPO Guidelines for Examination, 'Part G: Patentability, 3. Person Skilled in the Art' (EPO, November 2019) <https://www.epo.org/law-practice/legal-texts/html/guidelines/e/g_vii_3.htm> accessed 9 March 2020.

2.5. Software Exhaustion

2.5.1. UsedSoft v Oracle

In 2012, the Court of Justice of the European Union (CJEU) made a decision in the landmark case *UsedSoft v Oracle*.[80] In this case, the CJEU laid down conditions in which the download of a computer program could constitute a first sale and thereby trigger exhaustion of distribution rights per Article 4(2) of the Software Direc-tive, even if the denominator of the agreement was a licence. Such an occurrence would constitute a first sale if the acquirer was allowed use, unrestricted in time and scope, of an object, tangible or intangible, in return for a payment that corresponded to economic value. This confirmed that intangible objects can be property and that a download can be seen as a sale.

How the findings fit in with FOSS is yet to be established. The dynamics of the agreement between the author and licensee (or possibly the first acquirer, if the title of first sale is indeed attributed) are slightly different. In the case of FOSS, the program is not readily provided as a down-load – rather, the code is provided. Moreover, more often than not, the software is available for free. The CJEU has placed emphasis on both the perpetuity of the agreement and a remuneration corresponding to economic value.[81] The former of these conditions would be fulfilled, as the very nature of FOSS is that a licensee is able to do with the software as they wish. The latter condition is not as certain, as there often is no renumeration. It could be argued that, because FOSS is available for free, its economic value is zero. However, the estimated value will likely exceed zero, as the wording of the CJEU is:

> *'which enables the copyright holder to obtain a remuneration equal to economic value'*

which implies that it is not the actual realisation of the remuneration that is of importance, but rather the mere possibility of obtaining remuneration.[82] The fact that the author has waived this and released the software free of charge would be inconsequential. That would mean that a FOSS licence could in fact constitute a first sale under Article 4(2) of the Software Directive. However, this has not been confirmed by the CJEU, so it remains to be seen whether or not and in what manner the definition of a sale might be applied to copyleft licences.

2.5.2. Consequences of Software Exhaustion for FOSS

Assuming that the contractual agreement when obtaining FOSS would qualify as a sale, this would have two important consequences for copyleft licences, both of which are destructive to their nature and continuing existence. First, recall that a copyleft licence requires that subsequent distribution of software, whether modified or not, be subject to the same licence it was initially acquired under. However, if the initial transaction is qualified as a sale, this obligation is no longer compatible with the agreement. In fact, the CJEU defined a sale as:

> *'an agreement by which a person, in return for payment, transfers to another person his rights of ownership in an item of tangible or intangible property belonging to him'.*[83]

This definition should be used uniformly and ubiquitously throughout the EU, as the legislation makes no reference to national legislation.[84] The definition given speaks of a complete transfer of rights of ownership. Therefore, the previous owner would be in no position to oblige the subsequent owner to further distribute the software only under certain circumstances. Indeed, the purchaser would hold the rights to the copy of the software and could distribute it further as they please, as any other situation would be incompatible with the nature of a sale.

Second, to promote the free movement of software, copyleft and permissive licences provide the user with the right to modify software and then distribute the modified software. However, in the case of a sale, such rights would also fall away. A purchaser is not allowed to modify and distribute the software without a licence from the copyright holder, because Article 4(2) Software Directive specifically and exclusively exhausts the right to distribution. Consequently, the further distribution of modified software without a licence would constitute copyright infringement of the rightholder's exclusive rights under Article 4(1)(b) Software Directive.[85]

In summary, if the licence in the case of FOSS is seen as a contract of sale, the interests of both the author of the software and those interested in utilising it in some manner would be adversely affected. The benefits of using FOSS would be eliminated. Fortunately, it seems unlikely that the exhaustion doctrine of the Software Directive will be applied in this manner. A careful reading of *UsedSoft* gives a clear requirement that the first acquirer would need to make their copy of the software unusable to trigger exhaustion and circumvent infringement.[86] Two issues come to mind when applying this to a FOSS situation. The first is whether the sample of source code would qualify as a copy at all. Since there is no real transfer of an object from one person to another, is the case for tangible things, and since *UsedSoft* applies also for a downloaded program, it

is more likely that the publishing of FOSS in this case would be seen as a 'making available to the public' rather than 'distribution' and thus not trigger exhaustion. Second, whether or not a user has made their own copy unusable before a subsequent sale is impossible to guarantee in the case of FOSS. In *UsedSoft* or other proprietary software cases, the CJEU has granted that a copyright holder may make use of technical protective measures, such as product keys.[87] While this is difficult for digital goods in any case, it is in direct contradiction of the nature of freely available software. Granted, this is a narrow reading of *UsedSoft*, but such a view is supported by subsequent case law, which implies that this was indeed the intention of the CJEU.

2.5.3. A Nuanced View

In the 2016 case *Microsoft*, the question arose whether such exhaustion could extend to the backup copy that a first acquirer is allowed to reproduce per Article 5(2) of the Software Directive.[88] The conclusive answer from the CJEU was no: the backup copy cannot be sold and is meant purely for personal use, thus already providing a limit to the exhaustion principle. Moreover, it is emphasised in *UsedSoft* that the judgment only applied within the context of the Software Directive, which is lex specialis. Ironically, it is not always obvious whether or not software is governed by the Software Directive. In *Nintendo*, the CJEU stated that a video game was not governed by the Software Directive but rather by the InfoSoc Directive.[89] Although computer programs were the composing elements of the work in question, they were not its substance, as it was a complex work. This despite the fact that the creative elements, such as graphics and sound, were necessarily encrypted in computer language.[90] If combined with the judgement from *Art & Allposters*, in which the CJEU concluded that exhaustion under Article 4(2) of the InfoSoc Directive was limited to tangible objects, this limits the scope of *UsedSoft* even more.[91]

Late last year, the CJEU made a decision in the case *Tom Kabinet*, which concerned the retail of 'used' e-books.[92] This provided clarity on whether exhaustion applies in the 'distribution' of digital goods and whether sales of such qualified as distribution. The first of four questions posed addressed distribution directly:

> *1. Does the making available remotely by download of e-books (digital copies of books protected by copyright), for use during an unlimited period, against a price which enables the copyright holder to obtain remuneration corresponding to the economic value of the work, qualify as 'distribution' in the meaning of Article 4(1) of the InfoSoc Directive?*

The CJEU decided that sale of "second-hand" e-books does *not* qualify as distribution, but rather as communication to the public, which is not subject to exhaustion under Article 3(3) InfoSoc. The Court thus confirmed that the InfoSoc Directive enjoys a more narrow definition of distribution than the application of the Software Directive as seen in *UsedSoft*. This is consistent with the general principle of proportionality in EU law and case law alike.[93] Consider the CJEU's words in *Laserdisken*:

> *'[The principle of proportionality] requires that measures implemented through Community provisions be appropriate for attaining the objective pursued and must not go beyond what is necessary to achieve it.'*[94]

As concerns this objective, the CJEU often refers to recitals for the teleological interpretation of a work.[95] In the case of the InfoSoc Directive, these goals include preserving and developing creativity in the interests of authors and consumers alike, protecting intellectual property in order to guarantee an appropriate reward for the use of works and to provide the opportunity for satisfactory returns on investment, and providing a rigorous and effective system of protection to ensure that European cultural creativity and production receive the necessary resources and of safe-guarding the independence and dignity of artistic creators and performers.[96] These aims, applied to a FOSS environment, support that a proportional reading of the exhaustion doctrine would be a narrow one. Another interpretation would undermine open source itself and thereby the work of the author and enjoyment of the consumer/user.

The CJEU has been seen to consider situations in their entirety, whether they be situations of transfer, such as in the *UsedSoft* case – where the downloading of the program and the subsequent licence agreement (now sales

[80] Judgment of 3 July 2012, UsedSoft, C-128/11, EU:C:2012:407.
[81] Ibid. para. 45.
[82] Ibid. para. 49.
[83] Ibid. para. 42.
[84] Ibid. para. 39.
[85] Ken Moon, 'Where Does Free and Open Source Licensing Stand in Europe?' (Lexology, 20 August 2013) https://www.lexology.com/library/detail.aspx?g=91b10f02-8ae0-4e2d-bd20-4bba0e4fcfd6 accessed 9 March 2020.
[86] UsedSoft (n 80) para. 70.
[87] Ibid. para. 79.
[88] Judgment of 12 October 2016, *Ranks and Vasiļevičs*, C-166/15, EU:C:2016:762, para. 43.
[89] Judgment of 23 January 2014, Nintendo and Others Box, C-355/12, EU:C:2014:25, para. 23.
[90] Ibid.
[91] Judgment of 22 January 2015, Art & Allposters International, C-419/13, EU:C:2015:27, para. 40.
[92] Judgment of 19 December 2019, Nederlands Uitgeversverbond and Groep Algemene Uitgevers, C-263/18, EU:C:2019:1111, para. 1–2.
[93] Eleonora Rosati, Copyright and the Court of Justice of the European Union (Oxford University Press 2019), 47.
[94] Judgment of 12 September 2006, Laserdisken, C-479/04, EU:C:2006:549, para. 53.
[95] Rosati (n 93) 57.
[96] Rec. 9–11 of the InfoSoc Directive.

agreement) were seen as an indivisible act – or considerations of copyright-protected works, such in the *Nintendo* case – where the video game was considered in its entirety, beyond its encrypted form. This indicates that FOSS will also be viewed broadly, which means that its characteristics might differ depending on if the Software Directive or the InfoSoc Directive is applied. Any type of program can be FOSS, whether it constitutes a simple function not exceeding a few rows of source code or an entire video game. Which of the directives applies will have to be evaluated on a case-by-case basis. Since *Tom Kabinet* and *UsedSoft* arrived at contrasting decisions, this question will likely be the main issue in future disputes.

3. PATENTS

3.1. Issues Specific to FOSS

With copyright, an author is almost certain that the copyright of an original work is entirely their own. This means that when external software is used, you can assume that you are not trespassing on another's rights, as long as you comply with the licence conditions.[97] While this, as discussed, is not as simple as it may sound, it is still a lot simpler than the issue of patents. The two conditions that an author may rely on – originality and compatibility – will likely not protect them in situations with patent protection. This is specifically in contrast to the principles of FOSS.

First, this is due to the area of protection of patents. In copyright, if two authors have the same clever idea, they are very likely to have come up with different implementations/codes, which means they are not at risk for infringing each other's works. However, a patent will only be granted to one of the creators for the technical idea, which means the other cannot effectively use his or hers (different) code in question without risking patent infringement. This situation is in stark contrast to the nature of FOSS, which wants to promote widespread development and improvement and – most importantly – freedom in regards to software use.

Second, many patent-licensing schemes are quite different from copyright licences. Often, they entail running royalties and the obligation to report sales, which are contradictory per se to the freedom to make copies, distribute and modify the software as one pleases. Only royalty-free patent licences are compatible with FOSS – not licences that adhere to the FRAND terms.[98]

In summary, the possibility of patent protection for these types of inventions is actually detrimental to open source, because even if a user of FOSS complies completely with the licence it is released under, they might still unknowingly be infringing a patent which is granted for the idea which the FOSS is an implementation of. In fact, not even the author of the software usually knows whether their code is an infringement of a patent, as they can be hard to find – authors usually simply depend on the knowledge that their work is original. An author might not have the means to acquire a licence after the fact and obviously the publishing of source code makes proving infringement very easy for the patent holder.[99]

3.2. Possible Solutions

3.2.1. Third Party Patent Holder

In the situation where the patent holder is not associated with the FOSS, there are a few things the software developer could do. It is often argued that a developer can invent around an existing patent, in such a manner that their implementation does not touch the patent area. However, patent protection is often too broad and can be interpreted to encompass an entire problem rather than a solution. This means that it matters very little in what way a solution is phrased, as any solution to the same problem will be an infringement. Moreover, the patent, due to the broad formulation, often includes standards. It would not be possible to use such a standard without acquiring a patent licence.[100]

Another solution could be to make use of either shimming[101] or plug-ins[102] in a modular system. These are both ways in which the patented part can be embedded into a separate, patent-licence compliant part which merely interacts with other parts, which can then remain FOSS-licensed. It is, so to speak, a separation of interacting parts. However, neither solution is optimal – for two reasons. First, this unnecessarily increases the complexity of a program, and second – and most importantly – the FOSS cannot then implement a patented standard, but merely make use of it. Thus, the more such patents are granted, the more FOSS would shrink.[103]

The best thing to hope for, which would leave intact the disparate intellectual property protections of copyright and patents while simultaneously respecting the nature of FOSS and even software as a whole, would be that pure software could not infringe upon a patent which is granted for a CII or another type of software-embedded invention,

[97] Ibid.
[98] ibid. 43.
[99] Arnoud Engelfriet, 'Octrooirisico's bij Open Source Software' (Ius Mentis, 6 November 2018) https://www.iusmentis.com/computerprogrammas/opensourcesoftware/octrooirisicos/ accessed 9 March 2020.
[100] European Parliament (n 21) 47.
[101] Traditionally, a shim refers to a thin sheet of metal which one might use to link together two not entirely compatible parts, by filling the gap when the width or breadth of one component does not match the other. In programming, its function is similar. It is usually an API which translates signals from one part into signals that another can process. It might also be used to connect a patent-protected part to a FOSS program.
[102] A component that adds a specific feature to an existing program. It exists separately from the program, but interacts with it. A patent-protected program may be embodied in a plug-in so that it can be added to another program.
[103] European Parliament (n 21) 46.
[104] T-1099/06 (Transgenic plants cells/MAX PLANCK) of 30.1.2008, ECLI:EP:-BA:2008:T109906.20080130, Reasons for the Decision, p. 1-6.
[105] European Parliament (n 21) 48.

because computer programs as such are excluded from protection. Though teleologically consistent with the EPC, this is an entirely uncertain conclusion, made more so by the national nature of patents. As yet, national courts have the exclusive jurisdiction in patent infringement matters under Article 1 EPC. Moreover, the EPO ruled in *Max Planck* that there is no principle of binding case law in these matters.[104] This means that in the current absence of a Unitary Patent or binding Union-level case law, these matters are up to the sovereign nations and interpretations might differ greatly.

3.2.2. Patent Provisions in Copyleft Licences

If the party that holds the patent, but not the author of the FOSS, is involved in the distribution chain there are more possibilities. One of them is enclosing a patent provision within the licence. Existing provisions concerning patents most commonly take one of two forms. These are retaliation clauses and express patent-licensing clauses. The former entails that if a patent holder who receives the FOSS further distributes a certain computer program and then requires any of the recipients or the author to obtain a patent licence for that same program, the FOSS licence is terminated. This would mean that the patent holder had no right to distribute the program in the first place. This means that if patent infringement is claimed, there will be retaliation in the form of a copyright infringement claim.[105] Presumably, however, the author of the program would not be affected by this, because they grant the licence and thus are not subject to the terms for a licensee. There is a possibility that the author would then pursue litigation against recipients of the copyleft licence, claiming patent infringement. Most likely, such conduct would be prohibited by law as misleading practice. However, this is not entirely certain and pursuing litigation would add further complexity to an already complex situation, which would not support FOSS. It would be best if such a risk were avoided altogether.

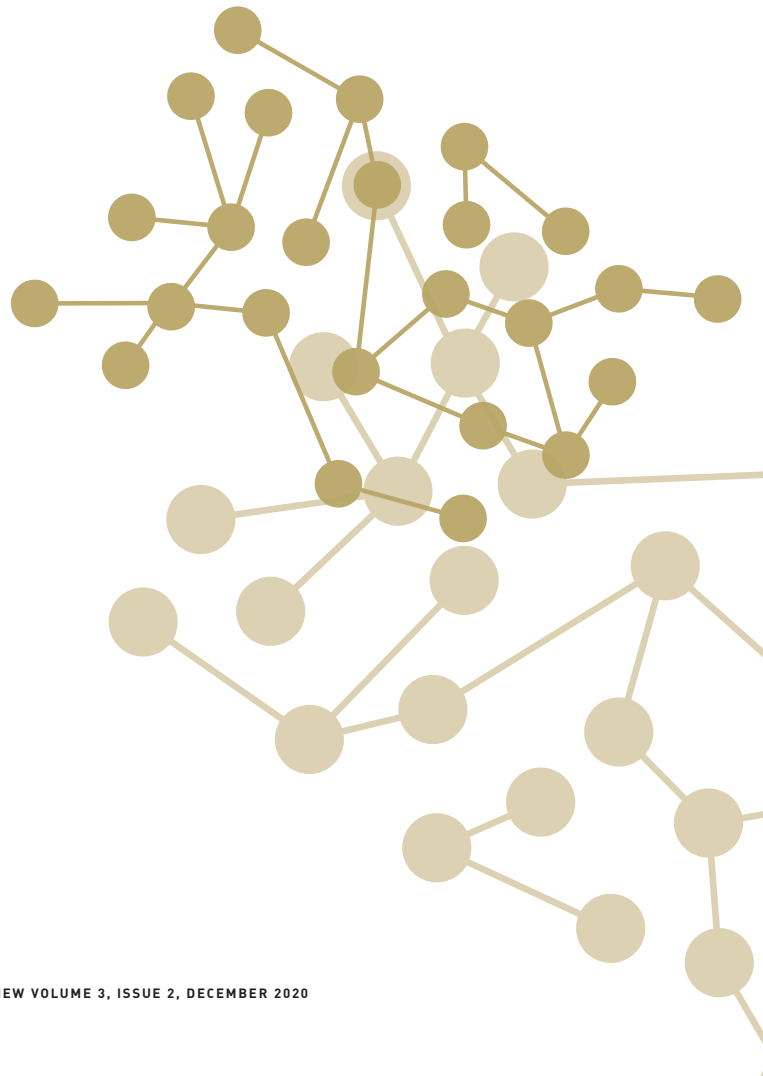
The other type of provision is the express patent clause. With the passing of time, more major FOSS licences have included a patent licence provision in their licence. Most often, this takes one of two forms. In the first, a patent licence is granted only in regards to the modifications that the patent holder has made to the program. Thus, another contribution in the same program, but not by the patent holder, might still trigger patent infringement. The MPLv2 includes such a clause in section 2.1., which states that:

> *Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:*
> *(2) under Patent Claims of such Contributor to make, use, sell, offer for sale, have made, import, and otherwise transfer either its Contributions or its Contributor Version.*

Although this is helpful in the sense that it eliminates the risk of misleading conduct on the part of patent holders associated with the program, it still gives no guarantee to a licensee that they are not infringing upon any patents. Considering the nature of software and patents, it is not unimaginable that a contributor might modify the program and thereby infringe a claim of the mentioned patent which is not covered by such a clause.

A second form of patent licence, such as in the GPLv3, is broader still. It covers all patents on distributed code, regardless of whether the patent holder was a contributor or merely received and distributed the code. The phrasing in section 11 of the GPLv3 is as follows:

> *'Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.*
>
> *If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.'*



Here, the contributor is not only responsible for their own patents, but also for patents they know to be valid in the relevant country, effectively targeting cooperation between third party patent holders and contributors. Moreover, the patent holder does not merely grant the licence for their contributions, but for any contributions relevant to the work. This grants more protection to the licensee, but might also be an undue burden for a contributor. It is feasible that a contributor, especially one in a corporate capacity, might be the licensee of multiple CII patents. Carrying the burden of downstream infringements on these patents would be a heavy responsibility. However, the licence does emphasise that the contributor has to 'have actual knowledge' than such downstream conduct would infringe any such patents. In the EU, this would probably translate to application of the test of the reasonable person, who knows or should have known.

3.2.3. Implied Patent Licences

Not all FOSS licences include express patent provisions – this is the case for most older versions. Here, the question arises whether such a licence should be taken to be implicit. In common law systems, this is the doctrine of implied licence. Many civil law systems of Europe have not established similar legal doctrine, but the following can be assumed to apply in analogy to the principles of silent or tacit agreement.[106] Naturally, whether or not this applies depends heavily on the licence itself and how it is formulated, but to show how this might be analysed, the BSD licence (which is permissive) and the GPLv2 will be used as examples.

In the BSD, there is no mention of a patent licence. However, it should also be noted that there is no express statement indicating that the licence is granted (only) under copyright. The same holds true for the MIT licence and the GPLv2, although the latter does limit the activities covered to distribution, modification and copying, which are associated with copyright.[107] Thus, the absence of explicit mentioning does not necessarily warrant the conclusion that patent rights are excluded from the licence. In fact, given that the right to 'copy', which is a quintessential copyright, is omitted from the licence and that there are no other provisions stating a grant exclusive to copyright might be argued to mean that the grant would cover all IP rights relevant to the program, including patents.[108] This conclusion might prove to be even more valid in the EU territory, in whose Member States the legal terminology might not correspond to the terms set out in the licence, which overwhelmingly originate in US law.

In section 7 of the GPLv2, the following is given:

> *'If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.'*

This does not explicitly state the grant of a patent licence, but any other conduct would be contrary to its phrasing. It results in a compulsory, royalty-free patent licence; not only does the contributor thereby licence their own patent, they are also obliged to obtain a licence for third party patents they are a licensee to, not only for the licensor but also for every downstream licensee.[109] This is consistent with the express patent licence in the GPLv3.

These examples further illustrate the difference between permissive and copyleft licences. In the case of a FOSS program acquired by a patent holder, who wishes to use that program and redistribute it, the BSD poses no problems. As it is permissive, the acquirer may simply opt to license the resulting product under a different licence and thereby protect their patent. However, for software licensed under the GPL, the acquirer is obliged to license a resulting product under that same licence. Thus, they would have to provide a licence *for every single downstream recipient*. This effectively negates patent rights within the territory of the GPL. If we read the FSF's words, this was presumably their aim.

> *'Every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary.'*[110]

In summary, it is likely that there is some type of implied patent licence in most FOSS licences. There certainly is in the copyleft licences, as concerns the patent claims that

can be read in the software. The exact terms depend on the wording of the licence, which could of course expressly exclude any patent rights. However, for the licence to be effective, it would be either illogical or purposefully misleading to exclude all patent rights, as this would make it impossible to use the source code without triggering it. Many licences do not contain a clause that permits use, modification and distribution only under copyright, which would be interpreted to permit use, modification and distribution in general and thus also in regard to potential patent rights.

3.3. Patent Exhaustion

If, as set out, exhaustion applies to a FOSS licence, and the author of the corresponding software further holds a patent which reads into said software, their distribution rights in regard to that copy of the software would be exhausted.

However, the EPC has no provisions on patent exhaustion. Additionally, it is not clear whether process patents – which CII patents nearly always are, as they encompass an idea rather than an implementation – can be exhausted and how that would impact the downstream distribution. Furthermore, the different courts in the EU might have very different interpretations.

Moreover, exhaustion usually only encompasses the use and distribution rights, not making new copies and distributing those. Then again, making a copy of a process is hardly possible. Even more than in the case of copyright, it is unlikely that the exhaustion doctrine would be applied in this context. Furthermore, it would not be a suitable solution, as it could not encompass modified downstream distribution and thus not guarantee risk-free conduct for the licensees.

All things considered, patent rights within FOSS are even more difficult to define than their copyright counterparts. This is unsurprising, as the licences often do not expressly mention patent rights – and nor do the commonly used licences originate in patent law. Lastly, there is no mandatory legislation at the EU level, and patents embedded in software are not mentioned, except in the exclusion from patentability for computer programs in Article 52(2)(c) EPC. National legislations can differ greatly, which is problematic considering the international nature of FOSS. Ideally, the Unitary Patent will be brought to life sooner rather than later and encompass a section on patents embodied in software, to clarify these matters. The corresponding legislation might interpret its section corresponding to Article 52(2)(c) EPC to mean that pure software cannot infringe upon a patent, as a patent on software is prohibited.

4. CONCLUSION AND RECOMMENDATIONS

"Open source is like Prison Break for developers, can we put a fence around this?"[111]

– Audience member at the Open Source Business Conference, 2010

The above quote raises a good question: can we define the boundaries of open source? The copyleft licence takes inspiration from US copyright law and reverses it. Instead of providing an author with the exclusive right to prevent use, reproduction, (re-)distribution, modification and communication, every recipient is provided with the 'inclusive' right to use, reproduce, (re-)distribute, modify and communicate the program under one condition: the program must remain free, so upon distribution the source code must be disclosed or available. This condition is where copyleft differs from permissive licences, which do not require this. Programs licensed under a permissive licence may be included in proprietary software without further requirements. In the case of the GPL – the original copyleft licence – this requirement goes even further by requiring that any reproduction, in whole or in part, modified or not, also be licensed under the same version of the GPL. The next question would be how we should define that work and, indeed, put a fence around open source.

Software is protected by copyright under EU law, denoting computer programs as 'literary works' under the Berne Convention. Thereunder, it is the source code as written by a developer that is protected, in consistency with a literary work – but less so with the nature of software, seeing as software serves a functional purpose, as a set of instructions for a computer to carry out. Due to this discrepancy, and in spite of Article 52(2)(c) EPC which excludes computer programs from patentability, there has been a rise in patents on software in the last few years. Companies that develop software, in one way or another, have an interest in utilising the vast amount of available open source code, because it has a low acquisition cost. This means the developers and the budget can be focused on qualitative innovation rather than having to start from scratch.

[106] Haapanen (n 18) 289.
[107] Ibid. 236.
[108] Ibid. 237.
[109] Ibid. 237.
[110] GPLv3, preamble, para. 9.
[111] Angie Hirata 'Top 10 Quotes from OSBC 2010 and What It Means for Open Source Developers' (ActiveState, 22 March 2010) https://www.activestate.com/blog/top-10-quotes-osbc-2010-and-what-it-means-open-source-developers/ accessed 9 March 2020.

"Open source isn't about saving money, it's about doing more stuff, and getting incremental innovation with the finite budget you have."[112]

– Jim Whitehurst, CEO, Red Hat

Ideally, the same company will want to avoid disclosure of proprietary source code as well as high-risk infringement suits. A copyleft licence requires disclosure in addition to distribution under the licence originally used; an obligation that encompasses derivative works, making the scope of that definition crucial. In light of the aim of the Software and InfoSoc Directives, as well as the nature of software, a derivative work should not be interpreted broadly. Interoperability is a key component of software and participation in the market requires compatibility with established software. Consequently, it would not be sensible to define a computer program as a derivative work for the sole reason that certain components enable interoperability. A suggested definition is 'a work based on an original work, in its entirety or in part, unless (a) the part in question exists exclusively to aid interoperability and promotion of a harmonised software environment and (b) the remaining connected software has been developed independently.' A useful tool to make assessments thereof would be to establish a legal fiction: a person who can read source code and software architecture as an average person can read text.

The effect of the digital exhaustion doctrine, which originated in *UsedSoft*, should also be interpreted conservatively. Subsequent case law shows that the CJEU supports such an interpretation. In the case of FOSS licences, the dispersion will likely not constitute a sale, due to a lack of protection or possible remuneration for the author in other cases. In addition, imposing potential measures to ensure compliance, such as encoded keys, would contradict the nature of FOSS. The application of exhaustion has differed between the Software Directive and the InfoSoc Directive, with the Software Directive supporting a stricter reading. This indicates that simpler code might be more easily exhausted, as more intricate computer programs have been regarded as complex works governed by the InfoSoc Directive.

Shifting the focus to patents, they protect the underlying idea of a computer program rather than its implementation, unlike copyright. Their reach within FOSS licences is the subject of controversy. This is unsurprising, considering that most FOSS licences neither expressly mention patent rights nor originate in patent terminology. Furthermore, the EU lacks harmonised legislation for patents in general and especially for patents on computer-implemented inventions. Fragmentation between national legislations is problematic, given the international nature of software.

Still, some FOSS licences do expressly mention patent rights. Some cases, such as the GPLv3, nearly eliminate a contributor's patent rights, while others, such as the MPLv2, do not cover downstream modifications,[113]. Finding a balance is difficult, because one wants to maintain respect for IP rights while simultaneously enabling the continuing existence of FOSS, specifically in a copyleft context. A task made harder by the lack of oversight regarding the quality, quantity and classification of patents on CIIs, which makes identifying infringing acts difficult.

As for licences that do not expressly mention patent rights, proper functioning of such a licence, at least in regard to software, requires that a licence is implied *if* the patent holder further distributes the program. Whether they have contributed to the program or merely distributed it should not be of consequence, as any other interpretation would incite misleading practices. However, such implied licences should not extend to patent claims affected by modifications of a downstream contributor who is not the patent holder, nor to requiring a licensee of a third party patent to provide all downstream recipients with a licence to that patent (except in incriminating circumstances). These conditions should only be possible by virtue of explicit terms.

Whether or not exhaustion should be applied in a FOSS context is inadequately substantiated. Patents on computer-implemented programs are predominantly process patents, under which it is illogical to speak of copies. Furthermore, exhaustion would not affect downstream distribution and modified versions of a program and is therefore an inadequate solution.

[112] Ibid.

[113] Mozilla Public Licence version 2.0 (Mozilla, January 2012) <https://www.mozilla.org/en-US/MPL/2.0/> accessed 9 March 2020.

[114] Many licences were last updated over a decade ago; the most commonly used ones (GPLv2, GPLv3, MIT) were all published over a decade ago.

Ideally, the Unitary Patent will come into force in the near future and be followed by clearer definitions and prohibitions concerning patents embodied in software. An interpretation of Article 52(2)(c) EPC in such legislation, showing that pure software cannot infringe a patent, as programs for computers are excluded from patentability, would be optimal.

Although there is a lot of uncertainty in the field of FOSS licences, cautious parameters can be formulated. The proper interpretation will vary on a case-by-case basis, as even the applicable legislation might differ, but a narrow reading of a copyleft licence is generally advisable, to ensure that companies can safely rely on open source, participate in the market and protect investments without risking infringement. For the same purposes and to uphold FOSS, relevant patent claims held by a downstream distributor or author that might otherwise be infringed should be regarded as being implicitly licensed upon distribution by said patent holder. Exhaustion is unlikely to apply in either copyright- or patent-related cases, but this remains uncertain until the CJEU has made a judgment on the matter.

The disorganised protection for computer programs under copyright in both the Software and InfoSoc Directives, as well as under patent law, needs to be addressed. Ideally, this would be done in a binding regulation at the EU level. It might be worth considering to depart from the current protective system and create a sui generis protective IP right for software which respects both its implementation and functionality, as was initially intended by WIPO in the 1980s. Otherwise, a directive or regulation on CII patents that addresses their interactions with copyright is necessary. In addition to conclusive legislation and case law, it might prove useful to establish a standard-setting organisation for FOSS licences, especially for the EU, which is home to many official languages and legal systems. This would promote the quality of such licences and ensure clear, EU-compatible terminology, as well as more frequent updates to counteract the neglect that many FOSS licences are currently subject to.[114]

Saar Hoek

Saar Hoek holds an LL.B. in Law from the University of Amsterdam (2017) and an LL.M. in European Intellectual Property Law from Stockholm University (2019). She is currently studying for an M.Sc. in Artificial Intelligence at Utrecht University.