To be published in the book
"Conference for the 30th Anniversary of the CRID"
(Collection du CRIDS)

Free and Open Source Software Licensing: A reference for the reconstruction of "virtual commons"?[1]

Philippe LAURENT
Senior Researcher at the CRIDS (University of Namur - Belgium)
Lawyer at the Brussels Bar (Marx, Van Ranst, Vermeersch & Partners)

The impact of the Free, Libre and Open Source Software (FLOSS) movement has been compared to a revolution in the computing world[2]. Beyond the philosophical and the sociological aspects of the movement, its most outstanding achievement is probably its penetration in the ICT market[3]. FLOSS has indeed reached a high level of economical maturity. It is not "just" an alternative movement any more, nor is it a simple option that companies of the field can choose to take or simply ignore. It is a reality that everyone has to acknowledge, understand and deal with, no matter which commercial strategy or business model is implemented.

FLOSS is based upon an outstanding permissive licensing system coupled with an unrestricted access to the source code, which enables the licensee to modify and re-distribute the software at will. In such context, intellectual property is used in a versatile way, not to monopolize technology and reap royalties, but to foster creation on an open and collaborative basis. This very particular use of initially "privative" rights firstly amazed… then became source of inspiration. Indeed, scientists, academics and other thinkers noticed that the free and open source system actually worked quite well and gave birth to collaborative and innovative, but also commercially viable ecosystems. They began to consider applying the same concepts to other sectors and tended to make of such peculiar open licences systems a reference for the reconstruction[4], the governing and the protection of new types of "commons".

According to a traditional definition of property law or classical economic literature, commons are "*a resource which all have a liberty-right to use, from which no one has a normative power to exclude others, and which no one has a duty to refrain from exploiting*"[5].

---

[1] The content of this paper has been presented in the workshop "Intellectual Commons: From Software to Biotechnology" in the framework of the Conference for the 30th Anniversary of the CRID, which took place in Namur from the 20th to the 22th of January 2010.

[2] CH. DIBONA, S. OCKMAN & S. STONE, *Open sources: voices from the open source Revolution*, O'Reilly Media, 1999, 288 p.

[3] On the economic impact of FLOSS, see for example the « Study on the Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU », prepared by R.A.GHOSH on November 20, 2006, for the European Commission, and available at http://ec.europa.eu/enterprise/sectors/ict/files/2006-11-20-flossimpact_en.pdf (May 2010).

[4] Even though this property law term "commons" dates back to long before the development of our "information society", the concept regains popularity, and is nowadays used in order to designate immaterial assets and resources that one tries to liberate from the Intellectual Property's yoke, hence the use of the term "reconstruction".

[5] S.R. MUZNER, "The Commons and the Anticommons in the Law and Theory of Property", in *The Blackwell Guide to the Philosophy of Law and Legal Theory*, M.P. Golding and W.A.Edmundson eds., Oxford, Blackwell Publishing, 2005, p.148. The term "commons" is also used to refer to the legal regime governing such shared

It traditionally refers to "open to all" natural resources such as grazing lands, fisheries or mining territories. In a broader sense, it has also been described as "*a resource shared by a group of people that is subject to social dilemmas*"[6]. "Open source", "open access", or "open content" are licensing schemes that have been perceived as a way to reconstruct virtual commons[7], or in other words, to mutualise assets that could otherwise be subject to exclusive rights or secretive behaviours.

The question that will be addressed in the present contribution is whether FLOSS licensing, considered globally and in its current state, corresponds to a way to reconstruct an immaterial "commons" and whether it can indeed serve as a reference. First, we will analyse the spirit behind the movement in order to identify its primary goal. Then, we will address two phenomena, which occurred from the natural evolution of the movement, and which paradoxically tend to make it drift away from these goals: the licences proliferation and incompatibility issue, and the highly controversial debate about the extent of the so-called "strong copyleft" effect.

I. Free and Open Source software: a unique goal

"*Extracting money from users of a program by restricting their use of it is destructive because the restrictions reduce the amount and the ways that the program can be used. This reduces the amount of wealth that humanity derives from the program. When there is a deliberate choice to restrict, the harmful consequences are deliberate destruction. The reason a good citizen does not use such destructive means to become wealthier is that, if everyone did so, we would all become poorer from the mutual destructiveness"* [8]. This statement quoted from Richard Stallman's[9] GNU Manifesto illustrates perfectly the social dilemma that lies behind any quest of "commons reconstruction."

Computer programs are immaterial and non-rival goods that could be freely reproduced, used, modified and redistributed by anyone. In a world where software would only be perceived as such, anyone could easily, and at no cost, benefit not only from the software, but also from the modifications and improvements that other users would bring to it and redistribute. However, software production requires knowledge, materials, time and energy,… in other words, investments. In order to promote the development of computer programs, to give developers an economical incentive to perform this task and to provide producers with a legal way to secure their investments, lawmakers decided to grant copyrights protection to computer programs. Software authors therefore benefit from exclusive rights, which allow them to control the reproduction, the communication, and to a large extent, the use of their original works. Alongside this legal means to regulate the use and diffusion of software, developers may also decide to grant or refuse access to its source code. Keeping source codes

---

resource. N. ELKIN-KOREN, "Exploring Creative Commons : A Skeptical View of a Worthy Pursuit", in *The Future of Public Domain*, L. Guibault & B. Hugenholtz eds., Kluwer Law International, 2006, p.33.

[6] CH. HESS AND E. OSTROM "Introduction: an overview of the knowledge commons", *Understanding knowledge as a commons, from theory to practice"*, Ed. Ch. Hess & E. Ostrom, Cambridge (MA), MIT Press, 2007, p.3.

[7] R. VAN WENDEL DE JOODE, J.A. DE BRUIJN & M.J.G VAN EETEN, *Protecting the virtual commons: Self-Organizing Open Source and Free Software Communities and Innovative Intellectual Property Regimes*, ITeR, TMC Asser Press, The Hague, 2003, 164 p.

[8] R. STALLMAN, *The GNU Manifesto*, available at http://www.gnu.org/gnu/manifesto.html (May 2010).

[9] Richard Stallman is the father of the Free Software movement and the founder of the Free Software Foundation (FSF).

secret is indeed an additional factual means that has been used by "proprietary software" developers to restrict and keep under fierce control the evolution of their programs.

The reaction against restrictive licences and source code secrecy resulted in Richard Stallman creating the GNU project. He also created the concept of "Free Software", which is defined by four freedoms that users must be granted. According to that definition, users must have the right to run the software for any purposes (freedom 0), to modify it (freedom 1), to copy and redistribute it (freedom 2), and to distribute modified versions (freedom 3). These freedoms imply that source code must be accessible[10]. The freedoms granted to users are really broad: even though no reference to intellectual property is made in their definition, they actually imply, in a legal system where copyrights automatically protect original works, that the author's rights must be licensed in a very permissive way to whoever receives the software.

"Free Software" is therefore defined by reference to the users, who are granted rights that are otherwise and normally under the legal exclusivity and, as regards modifications, the factual control of the programs' authors. In terms of "commons", by distributing software under a free software licence, and by providing access to the corresponding source codes, the author indeed (re-)creates a resource that can be shared and used by anyone. Furthermore, as software are infinitely reproducible and unlimitedly reusable without risk of scarcity or depletion, the exploitation of such reconstructed commons is immunised against some "tragic" effects that were analysed by Hardin with regards to commons in the limited world[11].

Also interestingly from the "commons" point of view, the third freedom summarises clearly the ultimate aim of the free software system: "*The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes*". In other words, free software licences do not only allow the creation of a shared resource, but also allow others to create new and additional resources that once again may be shared and re-used[12]. There is no limit to the exploitation of the resource in the Free Software definition, and, on the contrary, the more the resource is used, the more it is likely to actually develop, extend and improve[13].

Although Open Source Initiative (OSI) does not share all the philosophical aspects of the Free Software movement[14], its Open Source Definition, however, serves the same objectives. Whereas the ten criteria that a licence must comply with in order to qualify as "Open Source" licence[15] are much more detailed than the four Free Software freedoms, both qualifications refer to the same licensing scheme. Amongst these criteria, one could pinpoint the non-discrimination requisite as regards persons, groups or fields of endeavour. FLOSS is not supposed to be the exclusivity of a limited sector or community. It has a universal vocation.

So the main and global idea behind the FLOSS movement can be summarized as follows: one should let software freely expand, in terms of use, as well as in terms of production and

[10] "The free software definition", FSF official website, http://www.gnu.org/philosophy/free-sw.html (May 2010).
[11] G. HARDIN, "The Tragedy of the Commons", *Science*, Vol. 162, 13 December 1968, p.1263 *et seq*., also available at http://www.sciencemag.org/cgi/reprint/162/3859/1243.pdf (May 2010).
[12] Such further contribution is however not compulsory: any user may decide whether to redistribute or not his modifications.
[13] J. BOYLE, *The public Domain: Enclosing The Commons of the Mind*, Yale University Press, 2008, p.191 *et seq*., available at http://thepublicdomain.org/thepublicdomain1.pdf (May 2010).
[14] "Why Open Source misses the point of Free Software", FSF official website, http://www.gnu.org/philosophy/open-source-misses-the-point.html (May 2010).
[15] "The Open Source Definition", OSI official website, http://opensource.org/docs/osd (May 2010).

improvement of code. FLOSS seems therefore to correspond to the notion of reconstructed or virtual commons, as software programs seem to be treated as a resource that anyone can access and freely exploit without any limitation. However, this assertion is not totally correct in the practical field. The Free Software and the Open Source definitions are general principles that require implementation by way of licensing. Some leeway is left to licensors as to how to draft their licences. Furthermore, some limitations to the licences are also allowed and well accepted, which are however source of incompatibility amongst the different FLOSS licences and of disagreements amongst communities.

II. Proliferation and incompatibility of licences: the consequence of exclusive rights remnants

As a matter of facts, the drafters of FLOSS licences made the most of the leeway left by the definitions to draft licenses that fit their needs, objectives, philosophical standpoints and commercial strategies. Contrary to the original notion of "commons", which in its most traditional sense refers to resources that normally have "an existence outside of human activity"[16], or contrary to "public domain", which refers to goods that are not (or at least not any more) subject to any proprietary exclusive rights, the reconstruction of virtual commons by way of FLOSS licensing implies a specific dynamic. It departs from a created work, which is protected (or at least asserted as such), on which an author is granted exclusive rights by law, and which the latter would like to "put in the commons". In order to do so, he must exercise his exclusive rights and grant a permissive licence to any potential users. Paradoxically, such commons reconstruction is therefore achieved by taking the road of "private ordering"[17], namely the use of private prerogatives and *inter partes* agreements to create social norms. Even though the objective of the free and open source software movements is to liberate software from exclusivity, FLOSS licences are not just about granting rights or freedoms to the users. They also encompass clauses that more or less limit or condition the licence.

A. Some rights reserved… and more.

Even in the most permissive licences (the BSD[18] for instance), some clauses clearly reflect the will to nonetheless reserve some copyrights to the works. For instance, one of the basic provisions in any FLOSS licence is that a copyright notice cannot be removed from the source code. Reputation is an important asset in the FLOSS world, hence, developers usually insist on having their names displayed in the code or documentation and to be given credits when due. This corresponds to the exercise of the attribution (or paternity) rights that are granted to the authors by copyright laws, and whose aim is to protect the intimate bond that exists

[16] H. MITCHELL, *The Intellectual Commons: Towards an Ecology of Intellectual Property*, Oxford, Lexigton Books, 2005, p.70.

[17] N. ELKIN-KOREN, "What Contracts Can't Do: The Limits of Private Ordering in Facilitating a Creative Commons", *Fordam Law Review*, vol. 74, 2005. Available at SSRN: http://ssrn.com/abstract=760906 (May 2010); S. DUSOLLIER, "Sharing Access to Intellectual Property through Private Ordering", *Chicago-Kent Law Review*, vol 82:3, p.1391 *et seq.* ; S. DUSOLLIER, "The Master's Tools v. The Master's House: Creative Commons v. Copyrights", *Columbia Journal of Law & the Arts*, 29:3, p. 282.

[18] Berkeley Software Distribution licence (simplified version), available at http://opensource.org/licenses/bsd-license.php (May 2010).

between an author and his work. Even if the software is freed, this bond is willingly preserved. Such constraint is not meant to imply much compliance burden in terms of software: just keeping the notice with the name of the author in the headers of source code, adding information files in binary distributions, or spending a little more paper and ink when printed documentation is involved. However, the exercise may prove to be trickier in larger projects involving a lot of contributions and materials of different origins. Such attribution requirement is also not that innocuous in regards to the reconstruction of other particular types of immaterial commons. When it comes to scientific databases, for instance, even attribution requirements could constitute a burdensome restriction to the reuse of material[19].

Other classics in FLOSS licences are clauses which provide that no warranty whatsoever is granted, and that the licensor's liabilities are limited to the maximum extent allowed by the applicable law. The deal is easily understandable and is a matter of fairness: because the licence is royalty-free, the licensor does not want to be held accountable for any trouble. Such position elicits nonetheless some further considerations.

The existence of such disclaimers and, maybe even more, the shared feeling that they are necessary and useful, underline particularities of reconstructed commons. As the material originates from human activity and departs from a property regime, the mere fact of putting it in the commons and sharing it with others generates certain liability risks against which contributors feels a need of protection. In a private ordering scheme, this entails adding clauses to contracts. Adding disclaimers in software licences as regards the nature, the use and the effects of the software is directly borrowed from "proprietary software" practice, and tends to confirm the mass market licensing conception stating that "license is the product"[20]. Furthermore, it addresses personal commercial concerns of the licensor that are extraneous to copyright licensing in the strict sense, and so adds a contractual and therefore interpersonal layer to the licence[21].

Some licences, such as the MIT License[22] or the EPL[23], disclaim any non-infringement warranty. Given the current international situation of the software patenting issue, a clause providing that the licensor does not provide any warranty for third parties' patents infringement seems an unavoidable necessity[24]. However, licences diverge as to whether the licensor should warrant ownership of any intellectual property related to his contributions to the work. The MPL[25] provides a "third party claim", as well as a representation[26] stating that

---

[19] M. DULONG DE ROSNAY, « Check Your Data Freedom: A Taxonomy to Assess Life Science Database Openness », *Nature Precedings*, 2008, p. 5, available at http://dx.doi.org/10.1038/npre.2008.2083.1 (May 2010).

[20] R. W. GOMULKIEWICZ, "The License is the Product : Comments on the Promise of Article 2B for Software and Licensing", *Berkeley Technology Law Journal*, Vol. 13:89, 1998, p. 891 *et seq.* ; see also C. A. KUNZE, "Open Source/Free Software and the Section 109 of the Copyright Act", Testimony of Carol A. Kunze, on behalf of Red Hat, Inc., Hearing before the U.S. Copyright Office, Library of Congress and the National Telecommunications and Information Administration, Department of Commerce, November 29, 2000, available at http://www.copyright.gov/reports/studies/dmca/testimony/kunze.pdf (May 2010).

[21] M. J. MADISON, "Reconstructing the Software License", *Loyola University Chicago Law Journal*, Vol.35, 2003, p.298.

[22] Massachusetts Institute of Technology License, available at http://opensource.org/licenses/mit-license.php (May 2010).

[23] Eclipse Public License version 1.0, available at http://opensource.org/licenses/eclipse-1.0.php (May 2010).

[24] R. W. GOMULKIEWICZ, *op. cit.*, p. 906.

[25] Mozilla Public Licence version 1.1, available at http://opensource.org/licenses/mozilla1.1.php (May 2010).

[26] On the distinction between a representation and a warranty under US law, see for example L. DETERMANN, S. PIXLEY & G. SHAPIRO, "Managing commercial risks in open source software licensing", in *Journal of Intellectual Property Law & Practice*, 2007, Vol.2, n°11, p. 771 *et seq.*

the "*Contributor believes that Contributor's Modifications are Contributor's original creation(s) and/or Contributor has sufficient rights to grant the rights conveyed by this License*". The EUPL[27], on the other hand, includes a specific limited warranty from each contributor as regards the ownership of copyrights on his contributions.

Identifying the copyrights owners and adding disclaimers are reflexes borrowed from the proprietary software licensing tradition[28]. Furthermore, they seem to be a necessary practice imposed by the way business is done with FLOSS and by the current legal regime governing software. A lot of different additional clauses can sometimes be found in FLOSS licences, which are also influenced by the practices of the sector, marketing objectives or commercial considerations. They include licences on patent claims, obligations to mention that the software was modified and to document the modifications, reservation of trademarks rights, indemnification clauses, restrictions towards the use of the name of the contributors in advertisements, etc.

All these clauses do not serve the "commons" purpose, but the private interests of the contributors, sometimes to the detriment of the quality and homogeneity of the reconstructed commons. Whereas they are common practices in the software sector and have not impeded the progress of FLOSS, warranty and liability disclaimers, or any other additional clause, could also be much more problematic in other fields. For instance, mutualising life science databases[29] or biobanks[30] on a zero liability basis seems indeed much more unworkable. Therefore, if FLOSS licensing could be used as a model for the reconstruction of commons in other fields, each clause present in the licences should be carefully analysed, and its pertinence duly assessed, before being re-used or transposed.

B. Copyleft: protecting, fostering or altering the Commons?

Copyleft clauses are another very important type of additional provisions that condition some FLOSS licences. The FLOSS licences are traditionally divided into two main categories, namely copyleft licences and non-copyleft licences, depending on whether they include copyleft clauses or not. The copyleft mechanism has firstly been implemented by Richard Stallman in the GPL licences[31]. It conditions the licence to a reciprocity (or "share alike") principle which generally impedes the reuse of the covered code in a proprietary way. On the contrary, non-copyleft licences (like the BSD for example) generally allow the reuse of the covered code in proprietary software without too much restriction.

The first aim of copyleft is to make certain that the software and its modified versions remain FLOSS. This objective is reached by adding clauses in the licence providing that the licensees may redistribute the code or any program derived from it, on the condition that this

[27] European Union Public Licence version 1.1, available at http://ec.europa.eu/idabc/eupl (May 2010).
[28] A. SINTCLAIR, "License Profile : BSD", *to be published* in the *IFOSS L. Rev*., 2010.
[29] M. DULONG DE ROSNAY, *op. cit.*
[30] PH. LAURENT and L. VILCHES ARMESTO, « La constitution, la propriété et l'accès aux « biobanques » sous l'angle de la protection juridique des bases de données : place à l' « open access » ? », *Open science et marchandisation des connaissances*, collection *Cahiers Droit, Science et Technologies*, Vol.3, 2010, p. 193 *et seq.*
[31] The GNU General Public License in its different versions is available at http://www.gnu.org/licenses/licenses.html#GPL (May 2010). When not specified otherwise, the reference used in this article is the GPLv2.

redistribution occurs under the same terms and conditions. When distributing an improved version of a GPLed software, the licensee has no other choice but to license such work under a GPL. Another objective sought by Stallman when drafting the copyleft clause in the GPL is "*to encourage free software to spread replacing proprietary software that forbids cooperation, and thus make our society better*"[32]. Indeed, Copyleft definitely accelerates the expansion of free software by obliging any distribution of modified software to be done under the same licence: thanks to this effect, further contributions to the program may therefore be added to the common software pool and be reused by the community.

Such greater spreading is reached at the expense of the licensee's freedom of choosing how to exploit and license his derivative works. On the one hand, one could therefore perceive copyleft as a freedoms reducer and wonder whether a resource that is available under such reciprocity condition corresponds to the notion of "commons" in its purest form. The exploitation of the resource is indeed constrained by the clause, and the resource is consequently somehow fenced by the copyleft effect. This fence keeps some users away, namely those who would like to create derivative works without wanting, being able or even being allowed (*cfr. infra*) to redistribute them under the same licence. Some sort of exclusion therefore remains in the copylefted commons. On the other hand, copyleft clauses are a freedoms enhancer for the other users, as the latter are assured of benefiting from the same permissions in regards to redistributed derivative works. In other words, the licensor reduces the licensees' freedoms (first line of users) in order to ascertain that further users of the newly created versions of the work will benefit from the same permissive terms (second line of users). In that sense, the copyleft is presented as an acceptable rule as it "*does not conflict with the central freedoms*" but "*rather protects them*"[33]. Copyleft FLOSS is therefore an attenuated but fast spreading, auto-generating and "protected" type of reconstructed commons[34].

An important point which is worth stressing about copyleft is that this reciprocity scheme is allowed but not imposed by the Free Software or Open Source definitions. These definitions only provide for minimal user's freedoms, and not for any restriction as regards further licensing. The drafters of FLOSS license are therefore free to make their licence copyleft or not. They also have the liberty to shape their copyleft effect and to define its extent as they see fit, as long as this effect does not tamper with the minimal freedoms. As a result, a multitude of licences sprouted, creating a spectrum of copyleft effects of different extents and natures. All these different types of "copyleft" generally answer special needs of the licences' drafters, whose philosophies, business agendas or technological sectors differ from one another.

For systematisation and informational purposes, some copyleft classifications are put forward, on basis of the strength of the effect. In the GPL family, the GPL is considered a "strong

[32] R. STALLMAN, "Copyleft : Pragmatic Idealism", in *Free Software, Free Society: Selected Essays of Richard M. Stallman*, Boston, GNU Press, 2002, p.81, available at http://www.gnu.org/philosophy/fsfs/rms-essays.pdf (May 2010).
[33] "The free software definition", FSF official website, http://www.gnu.org/philosophy/free-sw.html (May 2010).
[34] Eben Moglen confirms that "*The goal of the GPL is to use copyright to create a "commons", a collection of resources to which anyone can add and from which anyone can borrow freely, but from which nothing can be permanently removed*" in his declaration of 26 February 2002 in support of MySQL AB, et al., in the case opposing the latter to Progress Software, Corp., et al. and brought before the United States District Court of Massachusetts. Available at http://www.gnu.org/press/mysql-affidavit.pdf (May 2010).

copyleft"[35], as it is based on the notion of derivative work[36] and operates to the fullest extent[37]. Any distributed derivative work based on the program must be licensed under GPL. The LGPL[38] is presented as a "weak copyleft" licence, as its copyleft effect is willingly limited. When applied to a library, LGPL allows the use of the latter by other programs (or the linking with other modules) without that the copyleft effect extends further than the library's code. It is therefore possible to create proprietary software that uses LGPL libraries. Other types of copyleft limitations exist. For instance, the copyleft effect of the Mozilla Public Licence (MPL) is limited on a "per file" basis: any file that encompasses MPLed code has to be available under the same licence. The other files of the program, as well as the latter itself considered as a whole, can be distributed under another licence. Some copyleft licences (the MPL, the CDDL[39] or the EPL for instance) allow also the distribution of object code under a proprietary licence, but only on the condition that source code remains available under the original FLOSS licence.

Copyleft effects can also differ depending on their triggers, namely the acts or uses that activate the obligation to re-license under the same conditions. Software distribution is the copyleft trigger for most of the licences. In contrast, Affero GPL licences[40] force the licensee to re-license the software under the same terms even if it is not distributed *per se*, but used on a server which is accessible by other users[41]. This type of copyleft has been devised by developers who create programs that are usually run on networks as "Software as a Service" (or "SaaS"). In that "cloud computing" context, the software is generally not distributed since service providers only deploy it on their own servers. Such providers may therefore use their improved versions of the program and allow other users access without having to actually comply with standard copyleft obligations. Affero copyleft has been created to fill this loophole.

The existence of such multiplicity of copyleft systems, which also entails their embodiment into different licences, makes us tend to nuance the abovementioned arguments that promote the acceptability of such reciprocity schemes as a measure that "protects" the central freedoms against appropriation. If one copyleft system, implemented by a unique FLOSS licence that is unanimously agreed on, governed the whole FLOSS commons, its fencing effect would indeed separate the commons from proprietary software, and impede the former to drift towards the latter. As different copyleft effects with different extents and triggers and

---

[35] On the use of the terms « strong copyleft» and « weak copyleft», one can refer to the webpage "Various Licenses and Comments about Them", FSF official website, http://www.gnu.org/licenses/license-list.html#SoftwareLicenses (May 2010). T. THALHOFER, "Commercial Usability of Open Source Software Licenses", in *Computer Law Review International*, n°5, 2008, p.130.

[36] *Cfr. Infra*.

[37] The GPL's strong copyleft has also implications as regards "collective works", which is a US law specificity that we will not tackle in this contribution. For more information on that notion, see for example M. WEBBINK, "Packaging Open Source", *IFOSS L. Rev.*, vol. 1, issue 2, 2009, p. 83 *et seq*. or L. DETERMAN, "Dangerous Liaisons – Software Combinations as Derivative Works? Distribution, Installation, and Execution of Linked Programs Under Copyright Law, Commercial Licenses, and the GPL", in *Berkeley Technology Law Journal*, Fall 2006, p.1421 *et seq*.

[38] The GNU Lesser General Public License in its different versions is available at http://www.gnu.org/licenses/licenses.html#LGPL (May 2010).

[39] Common Development and Distribution License, available at [39] Common Development and Distribution License, available at http://opensource.org/licenses/cddl1.php (May 2010)..

[40] The original Affero GPL version 1 is available at http://www.affero.org/oagpl.html, (May 2010) and theAGPLv3 is available at http://www.gnu.org/licenses/agpl-3.0.html (May 2010).

[41] This is the case in "Software as a Service" model, where software are not distributed but available on a server and usable remotely and on a "on demand" basis.

embodied in different and usually conflicting licences are currently in application, it causes a fragmentation of the commons, where fences act not only against appropriation, but also against the commons itself. In other words, it creates different types of "commons" or "sub-commons" that, in some cases, exclude one another. At this point, the question left to the assessment of the reader is whether a commons that is exclusive towards other highly (at least conceptually and teleologically) similar commons remains a true commons.

The use of a copyleft mechanism, the extent of its effect and the way it is activated are therefore questions of crucial importance in the creation of virtual commons. Even if, considered *in abstracto*, copyleft seems to allow the reconstruction of a "protected kind" of Commons, the use of such system has to be carefully considered as it could backfire against the commons itself. If upheld, it should be adapted to the type of the shared material and to the intellectual property that applies to the latter, and should be implemented in a way that is acceptable by maximum stakeholders[42]. Additionally, and maybe more importantly, one must also pay attention to the drawbacks of such forced reciprocity. The most important amongst them is the incompatibility problem that it generates when licences diversify.

C. Incompatible Commons?

With the expansion of the FLOSS phenomenon and the concurring development of technology, the licensing practice has also greatly evolved. As a consequence, FLOSS licences multiplied and diversified. More than 60 licences are currently listed as certified Open Source software by the OSI[43], and more that 80 licences are listed as free software licences on the FSF website[44]. As already illustrated above, they differ from one another not only by their drafting, but by the presence or absence of certain clauses or effects. The creation of new licences is usually a response to new needs, objectives or technologies, but also sometimes to new practice perceived by their drafters as threats to the FLOSS ecosystem. This is for instance the case of the Affero licences which reference the "distribution loophole" in the SaaS context[45]. The GPLv3 is also a good example of reaction against particular unwanted practices or behaviours, such as "Tivoization"[46], the use of digital rights management systems (DRM's), or software patent threats[47].

This diversity of licences reflects the diversity of business models that revolve around FLOSS, and must be respected as such[48]. However, this proliferation of licences has generally complicated the FLOSS landscape. Developers are more and more confronted with new clauses, new limitations or new conditions to fulfil. The development of projects based on and combining FLOSS of different origins increasingly implies burdensome and intricate compliance duties.

---

[42] On that regards, we refer to the contributions of the other participants of the workshop.
[43] See http://opensource.org/licenses/alphabetical (May 2010).
[44] See http://www.gnu.org/licenses/license-list.html#SoftwareLicenses (May 2010)
[45] *Cfr. supra*.
[46] This neologism refers to the practice consisting of using FLOSS on hardware but impeding the users to run modified versions of the software on that hardware by way of hardware restrictions (such as digital signatures).
[47] For an analysis of the new prohibitive features of the GPLv3, see for example A. ENGLEFRIET, "Uit Principe: de GNU General Public License (GPL) versie 3", *Computerrecht*, 2007, p.146 *et seq*.
[48] L. ROSEN, *License Proliferation*, 2005, p. 4, available at http://www.rosenlaw.com/LicenseProliferation.pdf (May 2010).

In the worst case scenario, compliance with each one of the licences applying to the combined codes is impossible, due to contradictory clauses. This gives rise to incompatibility situations. This "incompatibility" may indeed be described as a situation in which a licensee cannot comply with all the provisions of two licences that apply on two portions of code that he would like to merge or combine in a project. This means that if he merges or combines these codes together, he would be exposed to claims or sanctions for copyright infringement and/or breach of contract. The practical consequence is that the licensee must give up the idea of using at least one of the two codes in his project, even though he remains entitled to use them separately. On the contrary, compatibility refers to situation where two codes under two different licences can be merged or combined together in a project. In that circumstances, a special attention should however be paid as to the licence under which the result must be distributed. Indeed, compatibility does not mean that the choice is always left to the licensee as regards the licence under which the result should be distributed[49].

As explained before, Copyleft is indisputably the main source of incompatibility, and by definition, a copyleft licence tends to regulate more than the existing code: it also wants to govern further additional code. Incompatibilities arise when such additional code is already governed by a contradictory licence. Usually, this contradictory licence will be another copyleft licence, or a proprietary licence, but it is not always the case. The Apache licence[50], for example, is a non-copyleft licence that is incompatible with the GPLv2 because of the way some of its clauses are drafted and also because of the presence of other peculiar clauses such as indemnification and patent termination provisions.

The more additional code the copyleft licence wants to govern, the more situations of incompatibility are likely to happen inside a development project, and the more difficult it will be to circumvent the problem. This is the reason why "strong" copyleft licences generate the most troublesome incompatibility situations.

Some solutions have been devised in order to solve some incompatibility problems between copyleft licences. One of them is to add compatibility clauses (such as in the EUPL[51] or in some CECILL licences[52]), which specifically allows software to be redistributed under other specified copyleft licences in certain circumstances. The main advantage of such a system is its straightforwardness: there is no doubt as to whether the licences are compatible or not, and as to which licence will apply to the result. One of the drawbacks of such a system is that, by definition, the bridge created by such clause will only work towards some specific licences, and not towards any other licence. It implies, amongst others, that the compatibility clause (or the compatible licences list it refers to) must be revised when compatibility with other licences is wished.

In order to solve some compatibility problems, clauses of the GPLv3 provide for some flexibility towards acceptable "additional permissions" or "not permissive additional terms". These clauses have however a quite limited effect in practice, which is nonetheless at least twofold. It solves some compatibility issues that were identified between the GPLv2 and non-

[49] When combining code under GPLv2 and code under BSD, the resulting derivative work has to be distributed under the GPLv2 licence for example.

[50] Apache License version 2.0, available at http://opensource.org/licenses/apache2.0.php (May 2010)

[51] On the compatibility clause of the EUPL, see F. BASTIN and PH. LAURENT, "Report on the study of the compatibility mechanism of the EUPL v1.0", prepared for the European Commission Enterprise Directorate General on 11 September 2006, and available at http://ec.europa.eu/idabc/servlets/Doc?id=27472 (May 2010)

[52] The texts of the different CECILL licences in their different versions are available at http://www.cecill.info/licences.fr.html (May 2010).

copyleft licences (such as the Apache, *cfr supra*), and it allows people to use the GPLv3 even though they were considering the use of a more permissive licence (in that case, one only has to add "additional permissions" to the licence). Though, such system does not solve any compatibility problems that naturally exist with other copyleft licences. In that regard, it is worth underlining that even GPLv2 and GPLv3 are sometimes incompatible with each other[53].

Some other creative clauses have been devised in the artistic field. The version 1.3 of the "Licence Art Libre"[54], for example, provides for a general compatibility of the licence towards other copyleft licences that comply with some criteria, one of them being the existence of reciprocity. This implies that such other licences, in return, must also ensure compatibility with the Licence Art Libre. This quest towards reciprocity may be analysed as an exhortation for the different licences drafters to get closer and solve jointly the problem. Such movement is also promoted by Schmitz, according to whom the creation of a "circle of trust" between FLOSS licences stewards could be a possible way out to incompatibility problems[55].

Gathering all the communities together in order to discuss and solve commonly the FLOSS licences compatibility issue is indeed probably the best theoretical solution. In practice however, this entails the making of concessions that may tamper with strong philosophical stances, market positions or business factors, and that the stakeholders could not easily accept. Incompatibility problems in the FLOSS world not only illustrate the limits of the reconstructed commons, it also underlines some of the difficult and intricate aspects of its governance.

III. A strong copyleft… based on a strong(er) copyright ?

The extension of the GPL's "strong" copyleft effect through dynamic linking is probably one of the most controversial legal issues that have ever been debated about the licence. Dynamic linking is a specific manner of combining different programs. Before presenting what the debate is about, some technical background is required. The example of computer libraries will be used in order to illustrate the technical aspects of the issue.

A. Some technical notions

Four stages can be identified from the coding of a specific program until its execution. The first stage is the creation of source code, which is usually written by humans in a language that they can easily understand, and stored in multiple files. Before being treated by a machine, source code must usually be compiled into object code thanks to a "compiler program". In other words, files in source code are transformed, at the second stage, into files in object code. At the third stage, these files in object code are then linked together (by way of

[53] A "compatibility" matrix is available on the FSF website, which considers systematically all the possible or impossible combining situations. See http://www.gnu.org/licenses/gpl-faq.html#AllCompatibility (May 2010).
[54] Licence Art Libre 1.3, available at http://artlibre.org/licence/lal (May 2010).
[55] P.E. SCHMITZ, "Copyleft Licences Interoperability. (Reducing the Impact of proliferation)", *to be published* in the *IFOSS L. Rev.*, 2010.

a "linker program") to create an executable file. Run time designates the fourth stage, namely the moment when the executable program is loaded into temporary memory and processed by the CPU.

Dynamic linking is quite common when developing programs that use libraries. Libraries are about not reinventing the wheel: they are files that contain a multitude of useful pre-coded functions that can be reused in or by other programs. They are comparable to tool-boxes in which developers can pick up standard functions. For example, instead of re-coding a routine, a developer could decide to get an existing library that encompasses that routine and to use it in his project. Usually, the library is also a file that is coded in source code and compiled into object code. From that point, two main techniques exist in order to make the program work with the routine provided by the library: static linking and dynamic linking. The difference between the two is the result after the linkage phase.

When the linkage is static, a part of the library (the wanted routine) is reproduced and put along with the developer's object files to create a single executable file. In that case, the executable program file is self-sufficient and can be run without the presence of the library on its side (as the wanted routine has been integrated in the executable).

When the linkage is dynamic, the linker does not copy the content of the library into the executable file. It only adds references to the library's function, as well as a command to load the library into memory alongside the program. In that case, the program can only be run if the library file is also installed on the computer and loaded into temporary memory when the program is executed.

B. Variable or missing factors

What is the extent of the copyleft effect when making a piece of software "use" another? Whereas this question is essential, its answer is often complex as it depends on different factors, and sometimes leads to disappointing uncertainties. The answer depends first of all on the licence itself. As presented above, there is a multitude of licences with different types of copyleft effects. When GPL is involved, the debate fires up, as it raises an even more sensitive question: how strong is the GPL's "strong" copyleft?

Some situations are clear-cut cases. When GPL-licensed software is improved by editing it, or when extensive parts of it are copy-pasted into the source code of another program, there is little doubt that the result is a derivative work that is subject to the terms of the copyleft clause. At the opposite side, the mere fact of storing an independent program (that is not based on any GPL-licensed software) on the same medium as other programs that are GPL-Licensed does not trigger the copyleft effect[56]. There exist however a multitude of in-between cases requiring deeper analyses or involving a certain degree of ambiguity. Dynamic linking is the combination that is subject to the most divergent opinions.

In order to try to solve the dynamic linking case, one must first of all analyse the wording of the GPL. The licence allows the licensee to "*modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program*"[57] on certain conditions. The

[56] See article 2, last sentence of the GPLv2.
[57] GPLv2, art. 2.

GPL licence defines a "*work based on the Program*" as "*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*"[58]. The specific copyleft requirement applies to any work "*that in whole or in part contains or is derived from the Program or any part thereof*"[59]. Even if this language is not unambiguous, what seems to be the reference is the notion of "derivative work" under copyright law[60]. A second step is therefore to determine what copyright law is applicable. In the absence of a provision explicitly determining the applicable law, this is far from being an easy task in situations involving parties from different countries. Last but not least, once the applicable law is determined, one must then find out what a "derivative work" is and how it is regulated under such law. This is not an easy task either, as it seems that even if the concept of derivative work is usually acknowledged in national laws, they usually do not encompass a precise definition of the notion, and give even less clues as to how to apply it to software combinations. Furthermore, the notion of derivative work is intimately linked to the notion of originality, which is also a concept that varies from one country to another[61]. One must therefore conclude that there is unfortunately no unique answer to the problem, as it could change from one country to another[62], and that even in a given country, it shall usually be a matter of interpretation.

Instinctively, and regardless of the applicable law or case law, the copyright specialist can at least picture what is problematic or debatable in the dynamic linking case and why the Gordian knot is so difficult to cut. A derivative work is usually a work that "includes" at least an original part of a pre-existing copyrighted work. Such "inclusion" can only be done with the author's permission. The whole debate revolves around what sort of software combination implies such protected inclusion. Amongst others, the nature of software[63], the diversity of the possible combinations and techniques, and the constant evolution of the latter render the task complicated. Inclusions involving literal reproductions do not seem to raise a lot of questions. For example, authors from different countries or legal cultures seem indeed to agree that static linking creates derivative works under their national law, as a part of a pre-existing library is reproduced as such in the resulting executable software[64]. Things get complicated with

[58] GPLv2, art. 0.
[59] GPLv2, art. 2 b)
[60] M. VÄLIMÄKI, "GNU General Public License and the Distribution of Derivative Works", in *Journal of Information, Law and Technology*, 1/2005, Volume 10, Issue 1, available at http://www2.warwick.ac.uk/fac/soc/law/elj/jilt/2005_1/valimaki/ (May 2010).
[61] In Europe, the concept of software originality has been defined in article 1 of Directive 2009/24/EC on the legal protection of computer programs, which provides that software "*is original in the sense that it is the author's own intellectual creation*". Such criterion is applicable to any part of a work : in the Infopaq case, the European Court of Justice confirmed indeed that "*the various parts of a work thus enjoy protection […] provided that they contain elements which are the expression of the intellectual creation of the author of the work*" (Case C 5-08, 16 July 2009, § 39).
[62] On this regards, one will notice for instance that the EUPLv1.1 licence defines the notion of derivative work as "*the works or software that could be created by the Licensee, based upon the Original Work or modifications thereof*", but further states that "*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 extent is determined by copyright law applicable in the country mentioned in Article 15*". See EUPLv1.1, art. 1.
[63] B. CARVER, "Share and Share Alike: Understanding and Enforcing Open Source and Free Software Licenses", *Berkeley Technology Law Journal*, Vol. 20, 2005, p. 458.
[64] See for example R. PHILLIPS, "Deadly Combinations: A Framework For Analyzing the GPL's Viral Effect", in *Journal of Computer & Information Law*, 2008, vol. XXV, p. 498; L. DETERMAN, *op. cit.*, p.1497 ; L. ROSEN, "The Unreasonable Fear of Infection", 2001, available at http://www.rosenlaw.com/html/GPL.PDF (May 2010); M. VÄLIMÄKI, *op. cit.*; A. VAN HEKESEN & E. TJONG TJIN TAI, "Licentievormen van open source software", in *Open Source Software: Een verkenning naar de juridische aspecten van open source software*, Elsevier, 2006,

dynamic linking because in such case, a literal reproduction in the executable file does usually not occur. The library and the executable file remain logically separated, until they are both loaded in memory at run-time. However, even if a logical separation is maintained, the program cannot actually work if the library is not available on the computer. The inclusion is therefore primarily "conceptual". Many developments of technical or legal natures have been put forward in order to argue either that a program is a derivative work of the software it dynamically links with, or that the two elements are, legally speaking, separated works[65]. They include highly technical considerations on the dependency degree between the combined software (or their "intimacy"), the type of data needed and shared in order to carry out the dynamic linkage, the way they are loaded in temporary memory, the non permanent character of such loading, the distinctions between the different used memory spaces (kernel or user's)[66], etc... It would be too laborious and unsuitable to summarize all the positions and arguments in the present contribution.

We will just notice that even in the United States, from where most of the FLOSS licences including the GPL licences family originate, the answer to the dynamic linking question is far from being obvious, and divergent doctrines seem to divide authors[67]. What we propose to do however in the next point is to analyse the positions of two different licences stewards and their related communities on the issue, in order to draw conclusions as to paradoxical aspects of "strong copyleft" licensing.

C. Two licence stewards' standpoints

The dynamic linking debate has taken such proportions because of its multiple inferences. First of all, it pertains to one of the important specificities of the most used licences, namely the GPL licences. Furthermore, what is at stake is not only the precise extent of the GPL's strong copyleft, but also what explains the difference between the GPL and the LGPL (which encompasses some exceptions to the copyleft effect as regards linking with LGPL-licensed libraries). It is therefore an important element of the FSF's global strategy that is discussed.

The position of the Free Software Foundation (FSF), steward of the GPL family licenses, is clear and unambiguous: "*Linking ABC statically or dynamically with other modules is making a combined work based on ABC. Thus, the terms and conditions of the GNU General Public License cover the whole combination*"[68]. Also, according to the FSF, this rule not only applies

---

p. 106; M. BAIN, "Software Interactions and the Gnu General Public License", in *IFOSS L. Rev.*, vol. 2, issue 2, 2010, p. 165 et seq.; CHR. DE PRETER & H. DEKEYSER, "De tostandkoming en draagwijdete van open source-licenties", *Computerrecht*, 2004, p.34; T. JAEGER & A. METZGER, *Open Source Software - Rechtliche Rahmenbedingungen der Freien Software*, Verlag C.H. Beck, 2006, p. 44 *et seq.* ; M. HENLEY, "Open Source Software: An Introduction", in *Computer Law & Security Report*, Vol.24, 2008, p.82; B. JEAN, " « Option Libre » : compatibilité entre contrats ", 2006, available at http://optionlibre.eml.cc/OptionLibre_CompatibiliteEntreContrats.pdf (May 2010); PH. LAURENT "Logiciels libres et droit d'auteur : naissance, titularité et exercice des droits patrimoniaux", in *Les logiciels libres face au droit*, Cahier du CRID n°25, Bruxelles, Bruylant, 2005, p. 77 *et seq.*

[65] *Ibidem.*

[66] On these notions, see for example R. KEMP, "Current developments in Open Source Software", in *Computer Law & Security Repor*t, vol.25, 2009, p.575.

[67] See for example R. PHILLIPS, *op. cit.*, p. 498; L. DETERMAN, *op. cit.* p.1497 and L. ; ROSEN, *op. cit.*; M. L. STOLTZ, "The Penguin Paradox : How the Scope of Derivative Works in Copyright Affects the Effectiveness of the GNU GPL", *Boston University Law Review*, 2005, p. 1439 *et seq.*;

[68] "Frequently Asked Questions about the GNU Licenses", FSF official website,

to libraries, but to other linked software as well, such as drivers or plug-ins for example: "*If the program dynamically links plug-ins, and they make function calls to each other and share data structures, we believe they form a single program, which must be treated as an extension of both the main program and the plug-ins. This means the plug-ins must be released under the GPL or a GPL-compatible free software license, and that the terms of the GPL must be followed when those plug-ins are distributed*"[69].

That such rule is exclusively based on an interpretation of the applicable copyright law is also an important point to be stressed. One could indeed think that even if such extent of the copyleft effect cannot be reached by applying standard copyright law provisions pertaining to "derivative works", it could anyway be considered as a contractual obligation accepted by the parties. This would however miss an important point in the GPL licensing strategy[70], which is based on the postulate that, at least under US law, the licence is not a contract and involves no contractual obligations.[71] The drafting of the GPLv3 is particularly explicit on this regards when it defines modifying a work as "*to copy from or adapt all or part of the work in a fashion requiring copyright permission* [...]".

If upheld without any distinction or nuance, such rule would lead to impracticable or unreasonable situations. Operating systems generally encompass "shared libraries" that offer basic and common functions to any application that is installed on the computer. As soon as applications call upon such library, derivative works would be created, that would be governed by the GPL if GPL-licensed code is involved. This consequence is however inconsistent with the sector's traditions and practices, and would be totally unsustainable, as it would practically create a total cleavage between GPL and non-GPL environments (a GPL application could not run on a proprietary OS, and vice versa). This is why the GPL licences provide for an exception for system libraries excluding such elements of the copyleft mechanism[72]. The communities (and Linus Torvals[73]) also seem to agree that user programs are not derivative works of the Linux kernel[74].

This position of the FSF as to what constitutes a "derivative work" under US law can be compared with the Eclipse Foundation's point of view. The key notion of the EPLv1.0 licence is the notion of "contribution", as the copyleft effect applies only to such type of code. In the case of subsequent contributors (namely, licensees who are changing or adding to the original program), contributions are defined as "*changes to a program, or addition to a program,*

---

http://www.gnu.org/licenses/gpl-faq.html#LinkingOverControlledInterface (May 2010).

[69] "Frequently Asked Questions about the GNU Licenses", FSF official website, http://www.gnu.org/licenses/gpl-faq.html#GPLAndPlugins (May 2010).

[70] On the difference between "bare licences" and "contractual licences" under American and UK law, see for example M. HENLEY, "Jacobsen v Katser and Kamind Associates – An English Legal Perspective", *IFOSS L. Rev.*, vol. 1, issue 1, 2009, p. 41 *et seq.*; L. ROSEN, "Bad facts make good law: the Jacobsen case and Open Source", *IFOSS L. Rev.*, vol. 1, issue 1, 2009, p. 27 *et seq.*

[71] See for instance Eben Moglen's declaration in the MySQL case, *op. cit.*

[72] "Frequently Asked Questions about the GNU Licenses", FSF official website, http://www.gnu.org/licenses/gpl-faq.html#GPLIncompatibleLibs (May 2010).

[73] See http://lkml.indiana.edu/hypermail/linux/kernel/0312.0/0670.html (May 2010). Linus Torvalds also added a introductory note to the GPLv2 licence that has been used to license the Linux kernel, which reads as follows: "*NOTE! This copyright does *not* cover user programs that use kernel services by normal system calls - this is merely considered normal use of the kernel, and does *not* fall under the heading of "derived work". Also note that the GPL below is copyrighted by the Free Software Foundation, but the instance of code that it refers to (the Linux kernel) is copyrighted by me and others who actually wrote it*". This note is of particular importance as regards the use by applications of the kernel's shared library "glibc".

[74] R. KEMP, *op. cit.*, p.578.

where such changes and/or additions to the Program originate from and are distributed by that particular Contributor". The licence further specifies that "*Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program*". There are therefore two cumulative conditions in order for an addition not to be considered as a contribution: to be a separable module and not to be a derivative work of the Program. Hence, from the double negation of the sentence (in order not to be a contribution, an addition must not be a derivative work of the program) one can draw the conclusion that a contribution that is a derivative work of the program is an addition that qualifies as a "contribution", which enters in the copyleft's scope. The circular character of such definition has already been underlined by Webbink[75].

The Eclipse Foundation provides interesting information as to how the terms "derivative work" is used and interpreted in the framework of the EPL licence[76] on its website: *"The definition of derivative work varies under the copyright laws of different jurisdictions. The Eclipse Public License is governed under U.S. law. Under the U.S. Copyright Act, a "derivative work" is defined as "...a work based upon one or more pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations, elaborations, or other modifications which, as a whole, represent an original work of authorship, is a "derivative work"." The Eclipse Foundation interprets the term "derivative work" in a way that is consistent with the definition in the U.S. Copyright Act, as applicable to computer software"*[77]. In response to the communities which deem that linking to their code automatically creates derivative works, The Eclipse Foundation answers that "*linking to Eclipse code might or might not create a derivative work, depending on all of the other facts and circumstances*"[78]. However, it further states that writing an Eclipse plug-in with 100% your own code to implement a functionality not currently in Eclipse, or in other words, merely interfacing or interoperating with Eclipse plug-in APIs (without modification), does not create a derivative work[79]. The Symbian Foundation, which recently made the choice of using the EPL for its entire platform, further clarifies that "*in general, it is intended that you can write new closed-source applications that link to Symbian platform libraries and closed-source plug-ins for Symbian platform frameworks*"[80]. It must finally be noted that contrary to the GPL, the EPL does not encompass any "system libraries exception".

In conclusion, it seems that whereas both the GPL and the EPL licences have a copyleft effect based on the notion of "derivative work", the Eclipse Foundation's and FSF's positions do not concur as regards the extent of the notion when it is applied to plug-ins. Furthermore, according to EPL users, and contrary to the FSF's licensing practice, linking to a platform's library does not seem to require the addition of a specific exception in the licence in order not to be considered as creating covered derivative works.

---

[75] M. WEBBINK, *op.cit.*, p. 91.
[76] Note that the EPL explicitly provides that it is governed by the laws of the State of New York and the intellectual property laws of the United States of America.
[77] Eclipse Public License (EPL) Frequently Asked Questions, question 25, available at http://www.eclipse.org/legal/eplfaq.php (May 2010).
[78] Idem, question 26.
[79] Idem, question 27.
[80] See the Symbian developer wiki available at http://developer.symbian.org/wiki/index.php/What_is_Open_Source%3F (May 2010).

D. A paradox: commons guardians strengthening copyrights?

As usual, behind the FSF's clear-cut standpoint on the interpretation that must be given to the legal notion of "derivative work" lies a specific strategy aiming at promoting the expansion of free software: "*Using the ordinary GPL for a library gives free software developers an advantage over proprietary developers: a library that they can use, while proprietary developers cannot use it.* [...]*By releasing libraries that are limited to free software only, we can help each other's free software packages outdo the proprietary alternatives. The whole free software movement will have more popularity, because free software as a whole will stack up better against the competition*" [81]. This statement has its merits, and convinced a lot of developers, but it has nonetheless to be moderated when analysed from a "compatibility" perspective.

However, it seems that, by promoting a debated and far-reaching interpretation of the legal notion of "derivative work", the FSF actually contributes in its own specific way to the extension of the scope of copyrights, which his founder otherwise highly criticises[82]. Indeed, adopting such position is conferring upon authors a broad copyright control on what is done with their software... a control that the lawmaker has probably not considered or even imagined when conceiving the general notion of "derivative work", especially as regards computing techniques,... and a control that could ultimately be used by other copyrights owners in other contexts to further restrict accesses to some markets or to further confine the use of software at the detriment of FLOSS software developers and users or of any other person.

The dynamic linking debate has also repercussions on the compatibility issue. As explained above, incompatibility cases arise when compliance with two or more licences applying to parts of a combination of codes is impossible, due to contradictory clauses. Incompatibilities are more frequently faced when a licence wants to govern additional code, which another licence applies on. The more additional code the licence wants to govern, the higher is the risk that incompatibility situations occur. Extending the copyleft effect through dynamic linking results in broadening the targeted additional code, and therefore multiplies the incompatibility situations. For example, the interpretative position of the FSF has for direct consequence that an MPL-licensed program cannot dynamically link to a GPL-licensed library. The FSF's statement quoted above is therefore not completely true: using the GPL for a library will not "*give free software developers an advantage over proprietary developers*". It will give GPL-licensed software developers an advantage over proprietary developers and over users of FLOSS software that is licensed under other copyleft, but also free software licences. In other

[81] "Why you shouldn't use the Lesser GPL for your next library", FSF official website, http://www.gnu.org/licenses/why-not-lgpl.html (May 2010).

[82] We refer on this regards to Richard Stallman's strong statements: "*In the copyright bargain, the government spends our freedom instead of our money. Freedom is more precious than money, so government's responsibility to spend our freedom wisely and frugally is even greater than its responsibility to spend our money thus. Governments must never put the publishers' interests on a par with the public's freedom […]If copyright is a bargain made on behalf of the public, it should serve the public interest above all. The government's duty when selling the public's freedom is to sell only what it must, and sell it as dearly as possible. At the very least, we should pare back the extent of copyright as much as possible while maintaining a comparable level of publication"*. See R. STALLMAN, "Misinterpreting Copyright – A Series of Errors", in *Free Software, Free Society: Selected Essays of Richard M. Stallman*, Boston, GNU Press, 2002, p.81, available at http://www.gnu.org/philosophy/fsfs/rms-essays.pdf (May 2010).

words, it participates in the creation of contradictory and incompatible "sub-commons", and lessens the global software combinations possibilities, including sometimes amongst free software. This also underlines the fundamental distinction to be made between the free software definition and the way it is implemented and interpreted in the different recognised free software licences.

The "dynamic linking" debate also illustrates how the will to highly protect a reconstructed commons with reciprocity clauses can finally lead to strengthen the "anti-commons" by exposing unexplored aspects of the exclusive rights that it is based on, and by interpreting them in an extensive manner for the sake of the commons' protection.

More generally and fundamentally, it also illustrates the different positions that could be taken in the debates over reconstructed commons governance. On this regards, one could underline the divergence of opinions amongst FLOSS evangelists on the question whether FLOSS needs to be protected or not. Contrary to Richard Stallman and the FSF, whose standpoint has been commented above, Eric Raymond, one of the founders of the OSI, thinks for instance that "w*e don't need the GPL anymore. It's based on the belief that open source software is weak and needs to be protected. Open source would be succeeding faster if the GPL didn't make lots of people nervous about adopting it.*"[83]

Conclusion

By considering the fundaments of the FLOSS movements and the official definitions of "Free Software" and "Open Source Software", one notices that the objective of FLOSS licensing seems indeed to create a resource which all have the liberty to use and which no one has a duty to refrain from exploiting. Talking about commons when referring to FLOSS, like communities and their leaders do, makes sense.

Studying The FLOSS experience teaches us worthwhile lessons about the mere concept of commons, which seems to simultaneously refer to objectives as regards the exploitation of resources, to a set of rules that are adopted to reach such objectives, and to the final result, namely the shared resources governed by these rules. If the FLOSS definitions seem to concur with the notion of commons as an objective, we observe some perturbations or alterations at the licensing level.

Free Software and Open Source Software definitions are indeed only general principles that need to be implemented by licences, which remain the expression of the exclusive rights of the programs' authors. The existence of a multitude of different licences reflects the fact that there are different philosophical, political and/or commercial ways to conceive FLOSS. These elements are sources of divergences as regards the governance of the commons. The authors' exclusive rights on the programs are therefore exercised differently in order to serve different strategies.

[83] This is a quotation from a Speech given by Eric S. Raymond at the Fórum Internacional de Software Livre in 2005 in Brazil. See also Federico Biancuzzi's article "ESR : We don't need the GPL any more" based on an interview with Raymond, available at http://onlamp.com/pub/a/onlamp/2005/06/30/esr_interview.html (May 2010).

One of the major differences between FLOSS licences is the presence or absence of copyleft clauses, and the extent of such effect. The copyleft effect has been devised in order to protect FLOSS against appropriation, and to foster its spreading. However, copyleft reduces the freedoms of the licensee, whose choices as regards the exploitation of the code are restrained. Furthermore, copyleft has a fencing effect that does not only repel property software developers, but also sometimes the users and developers of FLOSS that are released under other incompatible licences. The strongest the copyleft effect, the more likely the licence will create incompatibilities problems. The concept of "protected commons" is therefore not just a sub-division of the concept of "commons" but is also source of "incompatible commons".

This issue is exacerbated when the copyleft effect is so strong that it seems to push the boundaries of the exclusive rights it is based on, as illustrated by the debate on the effect of copyleft on dynamic linking. Indeed, in order to enhance the protection of the reconstructed commons and its expansive effect, we observe a tendency from some communities to strengthen the "anti-commons" by way of an extensive interpretation of exclusive rights. This result seems also to somehow perturb the coherence of the system, and to create distortions between the means and the end.

We presented in this contribution that licences incompatibilities (which originate from the ability that is left to licensors to adapt their licence to their needs, objectives and strategies), create the division of the commons into "sub-commons" that are exclusive towards one another. That conclusion is drawn from a hi-level and global perspective, which considers FLOSS as a whole. The question that such approach raises is how homogenous a commons must be to be qualified as such. The fact that some pieces of the commons cannot be combined with other pieces seems indeed problematic and incompatible with the notion and the objectives that lie behind it.

From another perspective, one could present that each specific licence corresponds to a specific commons, which is used by and foremost destined to a specific community. The question raised by this approach is how big the community that participates in and profits from the shared resources should be in order for the latter to be qualified as a commons.

Another closely linked question that has been raised by analysing the copyleft protection of FLOSS is how much freedom should be given to the users of the shared resource to qualify as a commons and what kind of burdens can be imposed to these users. In other words, how protected a commons could or should be, knowing that the more it is protected, the more it implies burdens imposed to the users, and the fewer the latter are likely to be.

Whether FLOSS licensing is a good way to reconstruct virtual commons, and whether it is a good reference for other sectors depends, amongst others, on the answers to these questions which eventually pertain to the definition of virtual commons itself. For an outsider wishing to reconstruct virtual commons in another field, the most interesting fact about FLOSS licensing is analysing how far it has evolved and has been experienced, in order to better define, taking into consideration all the specificities of the targeted sector, what the notion of reconstructed or virtual commons implies in such a field.