Software licences and free software

UOC
Universitat Oberta de Catalunya
www.uoc.edu

Index

Introduction

This module focuses on software licensing, and free software licensing in particular.

Often, the creator of a software program is not its user: given that the law grants certain exclusive rights to the creator of a program in relation to its exploitation, as we have seen in Module 2, the creator must ensure that the user is granted sufficient rights so as to be able to use the software to the extent intended by the parties.

Granting these rights can be done in two manners: assignment and license.

- An assignment is a transfer of rights in an exclusive and definitive manner. This is the closest analogy to "selling" the program as if it were a good.
- A licence is the permission to perform an act (in relation to the work), which without that permission would be an infringement of copyright or a related right. As we will see below, a licence may be exclusive or non-exclusive, and may include several conditions upon use.

Most EU Member States require formalities of some sort (usually a written document) for assignments or licenses to be valid or validly proven.

The copyright legislation of most Member States imposes certain obligations on the contracting parties on the scope of the transfer of rights (e.g. on limitations on the transfer of rights relating to forms of exploitation that are known or foreseeable at the time the copyright contract was concluded or on rules on termination of contracts). Such conditions vary from one Member State to another.

1. Software licenses

In this first section, we look at software licences as a whole, before entering into the core part of the module which is free software licensing. We will also mention software assignments, which in continental jurisdictions are usually written as irrevocable and exclusive licences.

1.1. Concept of software licence

A software licence is a document or legal "instrument" that establishes the terms and conditions whereby the author or owner of the exploitation rights to a computer program (the "licensor", "software rightsholder") authorises the use of the program by another person (the "licensee" or "user").

We often say that a software licence[1] is an agreement, as it consists of an understanding between two parties: the software rightsholder and a user. It is nonetheless hardly ever an agreement that is negotiated between the two of them personally. Normally, it is the licensor that unilaterally establishes beforehand the terms and conditions of the licence, as an "accession" agreement. The user does not or cannot negotiate with the licensor the conditions of the licence, but rather must merely accept or reject them.

(1)Standard software licences are often called "End User License Agreements" (EULA), such as those that one accepts when downloading and installing a software program from the net, or from a purchased CD. See, for example, Adobe licenses.

In practice, we tend to use the term licence in two senses: to refer both to the software licence agreement (for instance, we speak of "accepting the terms and conditions of the licence"), and to the permission, authorisation or right granted to the user to use the software by the owner of the exclusive author's rights thereto (in this sense, we also speak of "having the licence to use software").

In fact, this dual meaning of the term licence has practical consequences and has given rise to disputes. And the reason for this is that, in certain jurisdictions (United States, United Kingdom in particular), software licences can be either:

- A contract that, in addition to the terms and conditions for the use of the software by the user, may establish other accessory agreements, such as confidentiality obligations, liability, competent courts for resolving any conflicts derived from the licence, etc.

Supplementary content

This has impacts as to contract formation (e.g. the contract requiring acceptance) and interpretation.

- A unilateral statement by the licensor, authorising the use of the software by those meeting and respecting certain conditions and limits, in accordance with the applicable law on author's rights. In this case, the express acceptance by the licensee is not required; therefore, the licence may solely refer exclusively to the right to use the software (and should not reg-

ulate accessory agreements, except as a condition for the exercise of the rights granted).

The main object of software licences is therefore to set out the conditions that are set upon the use (exploitation) of a software program, thus the rights that the licensor grants the user in respect of the software (what they may do with the software) and the limitations and prohibitions that must be respected by the user (what they cannot do).

Licences also regulate such other aspects as the following:

- Number of copies and/or licences granted.
- Method of delivery and installation of the software.
- Possible period of installation and acceptance tests by the user.
- Duration of the licence (limited, extendible or undetermined).
- Price of the licence (free, single payment or periodic instalments).
- Warranty period.
- Liabilities and limitations of liability of the licensor.
- Governing law and competent courts in case of litigation.

In this module, we shall focus our analysis on standard software user licences, whether or not customised to the needs of the user. Although it would seem that we have left aside "development agreements" (whereby a programmer receives the commission by a client to develop software according to their specifications), we should bear in mind that these agreements tend to be accompanied by a software licence or assignment in favour of the client. That licence, accessory to the "development agreement", is subject to the matters explained in this module.

Both non-free ("proprietary") and free software is commonly distributed by means of the same legal instrument: the user licence. The differences between proprietary software and free software are evident in the terms of the user licences that are completely different, especially in terms of the rights that the rightsholder grants the user:

- Non-free software licences tend to restrict user rights as much as possible, reducing them to a limited permission to use the software and to make a backup copy. The user is prohibited from copying, modifying or redistributing the software, and is usually provided a single copy in binary code.

- Free software licences contain a wide array of freedoms for the user, such as the freedom to use, copy, modify and redistribute the software. The supplier also provides the source code or makes it available to the user. Section 4 below sets out a detailed analysis of the content of free licences.

Software and its physical embodiment or medium

Software (either immaterial or a work of the intellect) is distinguished from the medium in which it is contained (material good): a hard drive, diskettes or CD-ROM, flash card. An important matter which must be quite clear is that although users acquire a copy of a computer program subject to a licence,

they are only "buying" ownership of the medium (the CD-ROM or the DVD, for instance) and not the software. In respect of the software, they are only acquiring the right to its use (a user licence), once the terms and conditions of the licence have been accepted.

This means that the user may transfer or sell the medium, and, if it is still during the term of the user licence, this may include the copy of the software (don't forget the concept of exhaustion, whereby the distribution of a copy of the work on a medium terminates the rightsholders' right to control redistribution of that copy). Likewise, the purchase of the medium does not automatically imply having the right to use the program, as the user must first accept the licence.

1.2. Software assignments

As we have noted, an assignment[2] is another means of granting to a third party rights over a software program (or any other work of authorship). However, an assignment is definitive, more akin to a sale, in that the original rightsholder is basically transferring the property of all the rights, irrevocably, to the recipient (known as the "assignee").

[(2)]Assignments are common in bespoke software development contracts and freelance or consultancy agreements, whereby the supplier assigns all the rights in the created software to the client. "The supplier hereby assigns all rights, title and interest in the [results of the work] to the client, free of liens and encumbrances" is a typical clause.

In continental European jurisdictions the concept of assignment generally does not exist, and the means to achieve the same result is by granting the recipient an *irrevocable, exclusive, royalty free, worldwide license of all the rights in the work, for the maximum duration of rights*. Remember that the creator will always have certain moral rights in a work, and these cannot be assigned or licensed to third parties.

Assignments or exclusive licences may be accompanied by warranties and indemnities that we comment below, just like any other software licence or contract.

1.3. Legal and economic function of licences

Why is the user licence the legal means or "instrument" commonly used by software rightsholders to distribute programs to the users? Basically because they are an efficient way to manage the rights of the software owner, who retains ownership and control over the program while at the same time permits dissemination (whether or not for profit) among users. This is due to the particular nature of software:

- Technical: Software is an immaterial or intangible good of which multiple copies of the same quality may be obtained; it is modifiable, giving rise to derivate works and, indeed, evolves continuously and quickly over time.

- Legal: Software is the object of exclusive author's rights granted by law to its creator, who may authorise its use, copy, modification or distribution, having no legal obligation whatsoever to disclose the source code.

The legal and economic functions of software licences differ, depending on whether they are "traditional" non-free software licences or free software licences.

- Companies developing non-free software benefit precisely from the exclusive exploitation rights granted to them by author's rights legislation. Seeking to obtain the maximum economic return on their software, non-free companies usually base their business model on the commercialisation or "sale" of copies: the more sold the better. Therefore, non-free software licences are traditionally restrictive in terms of content and scope of the rights granted to the user in respect of the software (no copying or modifying, no redistribution, no renting) and are very protective of the exclusive "reserved" rights of the author. It is basically and merely a "use" licence.

- In free software, licences have the same function, but an entirely different purpose. They are used to grant rights and establish obligations, but not to reserve the exclusive rights of the supplier or to commercialise the largest possible number of copies, but to grant and guarantee the rights of the users to use, modify, adapt, improve and redistribute the software. If economic benefits are sought, it is not through restricting user rights, but usually following a different business model (e.g. providing services for the software).

Thus, somewhat paradoxically, software licences, which have traditionally been used to restrict user rights, are also an adequate means of guaranteeing the rights of the users of software via the free licensing model. Free software license restrictions are more "conditions" for the user to exercise the rights granted; conditions that do not seek to reserve the exclusive exploitation rights of the rightsholder, but to preserve his/her reputation and guarantee that all users may benefit from such freedoms, thus preventing possible attempts at appropriating the software. We will see below in Section 4 the mechanism and content of free software licences in more detail.

1.4. Legal nature and regulatory framework

As we have seen in this module, a software user licence is a legal instrument. What kind of instrument is it and to what laws is it subject?

Generally speaking, a software licence establishes an atypical legal relationship, *sui generis*, that does not fall within the traditional set of (commercial) relations understood by courts: a purchase, a lease, a gift or a service rendered.

- The **purchase agreement** consists of a transaction whereby the seller delivers something (and the ownership thereof) to the buyer, in exchange for the payment of a certain amount of money (consideration or price). However, precisely one of the main reasons for granting a software licence is to avoid any outright "sale" of the rights in the software – an assignment. A licensor maintains at all times his/her (intellectual) property rights of the software and control of its copies and distribution.

Supplementary content

In the case of shareware, demos or evaluation copies, the licence could be assimilated to a loan, although they cannot be equated entirely: what is assigned is not a thing but a right.

- A **loan or lease** consists of the temporary transfer by a lessor of the possession and right to use something in favour of a lessee who, in exchange, pays a certain amount of money (normally in the form of rent or regular payments) – or, in the event of a loan, for free – and who, at the end of the agreement, must return the item to the lessor. A software licence cannot be entirely assimilated to a lease or loan: in many cases, a licence is granted for an indefinite term, while leases necessarily establish a determinate term for using the leased property. And even in cases in which the software licence is granted for a determinate term, what the licensor is granting the user-licensee are limited rights to an immaterial good, which are must less than those granted under a lease.
  What's more, when a lease agreement ends, the lessee must return the leased property to the lessor. In the case of software licences, although the user is sometimes required to return the copy to the licensor at the end of the agreement, the user often does not return anything at all, but rather destroys, erases and/or uninstalls the program for good.

- A **gift** is where something is transferred for free to another party. Freeware and free software licences are usually granted gratuitously and in fact could be most closely assimilated to gifts (of a non exclusive right, not of the software). The indefinite right to use the software is permitted, free of charge. Also, gifts may be conditional (when something is gifted to someone, but in exchange the beneficiary must meet a condition) just like free software licence grants rights subject, for instance, to copyleft conditions.

- When software is adapted or tailored to the needs of the user, it may also be assimilated to the results of the performance of work or the provision of a service.

Thus, while establishing an atypical relationship there is no legal norm that would specifically and comprehensively regulate software user licences, as opposed to "classic" agreements that are subject to legal regulation in Civil Codes or statute law, a software licence may bear certain characteristics of each of these relationships. Depending on the circumstances of each case, a court could apply directly or by analogy the law applicable to that relationship to

the software license: e.g. from a sale-purchase or lease agreement, the warranties of title (or "peaceful enjoyment"), good condition and operability of the thing sold or leased; from a lease, the obligation to return a good or ceasing using it when the term expires.

In addition, there is the debate whether a software licence (a free software licence in particular) is a contract or mere permission. In the first case, the courts would apply a large body of legal provisions for assisting in solving any difficulties in interpretation or application of the licence contract. However contract law also requires formal steps to be taken to ensure formation of a valid and binding contract: an offer, acceptance and, in the UK, consideration (basically, payment of price or promise). Certainly in the case of free software licences, there is the difficulty that often there is no express form of acceptance, nor indeed easily identifiable consideration from the user to the licensor.

So a more favourable view would be to see a license as a mere authorisation, as mentioned in copyright law, whereby the licensor authorises (unilaterally, in the case of free software licences) the users to carry out determined acts (copying, modifying, etc.) with or without conditions. Thus the conditions are not contractual conditions but licence conditions, breach of which would terminate the authorisation and any further act restricted by copyright would be a breach of the licensor's copyright rights.

In any event, the application of the norms applicable to contracts and the aforementioned forms of agreements should not occur in a generalised manner, but for specific scenarios, applying them "by analogy" to resolve disputes derived from aspects that are either not regulated by the licence itself, that are governed by ambiguous or incomprehensible conditions, or if a clause of the licence is considered null for breaching an imperative rule.

> It has specifically been said that the norms on purchase agreements may be applied by analogy to standard (mass) software licences that are more similar to a purchase, due to their terms and conditions (fixed price, indefinite time), but cannot be considered such.

In the end, a software licence is governed, above all, by the terms set out in the licence document and agreed between the parties and by the general norms on obligations. And we must also consider that the copyright law does indeed often regulate, at least partially and indirectly, the possible content of a software user licence agreement, with priority over the application or non application of other norms, e.g. as to exclusivity, term and geographic scope.

Finally, in any case, software licenses shall always be subject to certain laws and other norms:

- Mandatory law: The norms that apply mandatorily to licensor and licensee whatever the licence says. If the licence agreement contains a clause that

is contrary to an applicable mandatory rule, e.g. of consumer protection for providing a warranty, this clause will be null and void.
- Dispositive law: Norms governing the relationship in cases where nothing else has been established in the licence agreement. For instance, if a grant of rights is not expressly established as exclusive, by law it is often understood that the licence does not grant any exclusivity.

1.5. Subjective elements: parties to software licences

Two persons or parties are involved in a software licence (individuals or legal persons), who are granted certain rights and obligations. These parties are, on the one hand, the **supplier-licensor** of the software, and, on the other, the **user-licensee**.

1.5.1. The supplier-licensor

The supplier-licensor is the person who grants the licence to the user to use the software, providing him/her a copy of the licensed software. As we have seen, the supplier-licensor tends to fix the terms and conditions of the licence unilaterally, certainly with mass market licenses, and the user-licensee merely accepts or rejects them (being unable to negotiate the content of any rights and obligations).

The supplier-licensor must have sufficient rights in the software, according to author's rights legislation, to be able to grant the licence. As we have seen in Module 2, those who are authorised to grant licences are:

- The author or group of authors of the software (the original owner of its exploitation rights). These may grant user licences insofar as the exploitation rights to the software have not been assigned to a third party.
- A subsequent owner of the exploitation rights.
- A person who is solely entitled to distribute the software (a distributor).

Several comments need to be made:

- **Legal capacity**: authors may grant user licences for their software provided they are of legal age, i.e., they have the legal capacity to contract. Usually, software developers that are underage need authorisation from their parents or guardians to validly grant a licence.

  As an exception, national copyright laws sometimes allow underage authors to grant licences themselves if they are independent, e.g. "older than sixteen, living independently with the consent of their parents or guardians or with the authorisation of the person or institution caring from them".

- **Multiple rightsholders**. Please refer to Module 2 on the cases of multiple authorship and ownership of rights: the rightsholders may be "joint", requiring the consent of all the authors, or the rights may be collective,

under the control of a single party who has supervised or compiled the work of others, such as an "editor".

- **Derivate works**. Remember that derivate and composite works – based on or including prior works by third parties – may only be licensed to third parties in accordance with the rights granted by the upstream licence on the prior work. If this licence does not allow relicensing or sublicensing, or redistribution in any form, then the new work may not be licensed at all to third parties.

  In this context, for example, permissive free software licences such as the BSD or MIT allow any form of relicensing. On the other hand, the GPL only allows redistribution of derivative or composite works ("collective", in US terminology) under the same terms (the copyleft obligations) and indeed does not allow sub-licensing of the original code, but grants a direct license from the licensor to each new user.

- **Owner of exploitation or economic rights**. When the creator of a software assigns or licenses to another person any exploitation rights on an exclusive basis, we talk of a derivate owner or rightsholder, who becomes the person capable of exploiting the software, including therefore adapting it and redistributing it to third parties under a new licence. These rights may also be acquired by inheritance (heirs) or legal provision (employers).

  Agreements as to contributions to free software projects sometimes are drafted as assignments of the rights to the project, and grant a licence back to the author to allow them to continue to develop or exploit the software separately. Freelance or software development contracts also tend to include an assignment or exclusive licence grant, so the client has ownership of the result of the commission and, for example, may grant licences to third parties.

- **Distributors**. Just as many manufacturers market their products through distributors (who resell them, for instance, in a given territory), it is also possible for the author or the owner of the exploitation rights to the software to decide to market the software through a network of distributors, such as OEMs (Dell, HP, etc.), who often incorporate the software "as is" in hardware products, devices or appliances. The distributor is bound by a "distribution agreement" whereby it is authorised in turn to issue end-user sub-licences – often in the form specified by the rightsholder – while the distribution agreement remains effective.

  Computer warehouses or stores, where consumer software may be purchased, are not usually "distributors", in principle, as they merely sell the medium (CD-ROM, DVD) containing the copy of the software. The user licence is subscribed later, directly between the "manufacturer" of the software and the user (e.g. with a shrink wrap licence).

Warranty of title or peaceful enjoyment

Should the licence be granted by someone without such rights, the assignment or license of the right to use the software would be illegal and null and void.

In such a case, the licensor would have granted a licence in violation of the exclusive rights of a third party who holds the exploitation rights (e.g. the author of a component included in a software package), who may bring legal action to cancel the licences granted

without their permission, prohibiting the use of the software by the user and holding the licensor liable for damages.

The breach of third party rights by the software supplier in principle does not imply any responsibility for the user-licensee in good faith (i.e., a party taking the licence in the belief that the supplier was truly authorised to grant it). Nonetheless, the user may indeed sustain serious harm, specifically the suspension and loss of the right to use the software, as a result of claims or court actions being brought by the true owner. In these cases, the user will have also paid the wrong person for the use of the software.

Software copyright law itself provides that those who, without the authorisation of the owner, "place in circulation one or more copies of a computer program, knowing of or in a position to presume their illegitimate nature", are deemed in violation of author's rights (Article 7, EU Computer Programs Directive).

Therefore, having sufficient rights to grant a software licence is an intrinsic and *sine qua non* condition for doing so; and anyone granting a licence without having sufficient rights to do so will be liable to the user for any damages they may sustain if they are determined to have acted without sufficient authority to grant a licence. Therefore, it is said that in granting a user licence the supplier must necessarily grant the user a "*warranty of ownership*" or "*peaceful enjoyment*", whereby users are assured that they may use the software legally and that they may continue to use it for the duration of the licence.

In certain jurisdictions, more than a warranty, ownership of rights in the software is an inherent condition that the supplier must have over the software so that the licence is valid and does not infringe upon third-party author's rights. We nonetheless speak of "warranty of title or ownership" by influence of the law of English-speaking countries, as many software licences are a translation or adaptation of United States licences. Warranty of title: the supplier guarantees that they have the due authority to grant the licence and that no third-party rights are being infringed upon.

Additionally, should the user be a consumer, Consumer Protection Law applies, as we note below, whereby the supplier of software will be liable to the consumer user for the origin, identity and suitability of the software (often, in practice, for both consumers and independent professionals). Under these laws, clauses seeking to limit or exclude such warranty of ownership are generally null and void.

Many software licences follow the model of English-speaking countries of not granting any warranties on the software, not even a warranty of title, stating that the software is delivered to the user "as is". Many even expressly state that they provide no warranties of title and non-infringement.

As mentioned earlier, this exclusion of the warranty of ownership is probably invalid in most EU jurisdictions, as the software supplier is required by law to guarantee ownership of the software. It should be noted that the EUPL 1.1 (European Union Public Licence), drafted for the European legal framework, includes a "warranty of title", as does the OSL 3.0 (Open Source License). These licences are discussed below.

1.5.2. User-licensee

The user-licensee is the person acquiring the right to use the software under the licence, according to the terms and conditions established therein (almost always imposed by the software supplier). The main obligations of the user-licensee is to pay the price of the licence (when a paid licence is involved) and respect the user limitations imposed by the licence.

> In the case where the user is a licensee of non-free software, in principle, they usually have few user rights (basically, to run the program, use the application and make one backup copy, if not already provided), while the limitations are many. On the other hand, if the user is a licensee of free software, the freedoms of the user-licensee are much greater and, accordingly, the limitations are lesser: they could use the software freely, and modify and redistribute it, with or without modifications.

Should users be authorised to modify the software and they do, they may become the author of derived work (i.e., of the translation or adaptation of the software), as we have seen in Module 2. Additionally, if a user is authorised to redistribute the software and does so, they too may become software suppliers. This is often the case in free software development.

It is relevant to determine, in a software licence, whether the user-licensee is a consumer or a business, inasmuch as the legal system governing the licence and the legal norms applied to the relationship may vary accordingly, especially, in terms of the validity, application and interpretation of its clauses (for instance, regarding the termination of the agreement or the responsibilities of the supplier).

> Sometimes, the text of the user licence itself contains different rights and obligations depending on whether the user is a consumer using the software for personal use or a business using the software for its business activity. Much "freeware" or "shareware" is granted freely for personal use and subject to payments for business.

Consumers

If the user is a consumer, they are deemed to be in a particular weak negotiating position, which means that they have legal protection with respect to possible abuses by the software supplier. In this case, the licence is subject to the rules of the Consumer Protection, harmonised to a certain degree throughout the European Union, which prohibits abusive clauses. These are provisions that are not individually negotiated and that, contrary to the requirements of good faith, cause a significant imbalance to the detriment of the consumer of the rights and obligations of the parties.

Abusive clauses

Examples of clauses that are prohibited from being included, for being considered abusive:

- Clauses conditioning the performance of the licence to the unilateral will of the supplier: e.g. the right of the supplier unilaterally to construe or modify the terms of the licence, after its acceptance by the user, or freely to resolve the agreement, without notice or indemnification.
- Clauses stripping consumers of their basic rights: e.g. limiting or excluding the warranties that must legally be provided for the software and limiting or excluding their liability for damages caused by defective software.
- Other abusive clauses such as requiring the consumer to accept unknown clauses or conditions, forcing the user consumer to purchase unsolicited accessory goods or services or imposing that, in case of litigation with respect to the licence, courts other than those of the domicile of the user consumer should have the competent jurisdiction or that the licence should be subject to a foreign law, unrelated to the parties.

Additionally, when the user consumer acquires the software user licence over the internet (online), the software supplier must also meet the information obligations imposed by national implementations of the Ecommerce Directive[3]:

(3)The obligations under the national ecommerce law are imposed upon the licensors established in the relevant country. If the licensor is established outside of the European Union, the national (EU) ecommerce law shall solely apply to the licences granted to national consumer users and provided their web site is specifically directed to or has a specific section for that country.

- Before purchasing the licence, certain data must be provided in relation to the licence and the contracting process, in addition to the text of the general conditions.
- After acquiring a licence, the supplier must confirm with the user that their acceptance has been received and documentary evidence provided.

Business or professional users

Although the rules protecting consumers generally do not apply when the user is a business or professional, this does not mean that the software supplier may impose upon such clauses that are unfair or abusive. What in fact happens is that the business user does not have mandatory legal protection, whereby certain clauses are automatically deemed null and void.

However, a clause may be considered null and void, even in respect of business users, if it is considered to be contrary to the general rule of good faith that must govern the performance of the agreements, or a "reasonableness test" in the UK. This will depend on the examination of the circumstances of each particular case and in the end it is the courts who will decide whether the clause is contrary to good faith or unreasonable.

> Certain circumstances shall be considered as relevant when determining whether a clause should be annulled –for being abusive– when the licensee is a business or professional. For instance, whether the licensee is a large or small company, whether there has been a true process of negotiation between the parties, whether the user-licensee has accepted a clause that is unfavourable for its interests, in exchange for another favourable provision (for instance, a reduction in the price of the licence or a right to modify the software, in exchange for greater limitations to the liability of the supplier) or whether the supplier has simply imposed them.

1.5.3. Parties to a free software licence

In the case of free software licences, the traditional positions of supplier-licensor and user-licensee are maintained, however some specific points should be noted.

- First and foremost, the granting of a free software licence implies that its owner shares the exploitation rights with the users. This does not mean that the free software becomes part of the public domain, or that the rightsholders waive their rights. Free software is not software with "no owner". The author continues to maintain his or her status as author of the software and, in particular, maintains his or her moral rights in the work.

- By granting users the rights of to modify and redistribute the work, the user-licensee under a free software licence may, in turn, also become the supplier-licensor of other users, either by relicensing the same software (if they have the right to sub-license), or by licensing software derived from the original.

- Despite what free software licences often say – they tend to state that the software is offered "without warranties" – free software licensors must guarantee that the software does not infringe upon the right of any other software (whether free or non-free). The warranty of ownership and peaceful enjoyment is inherent in the condition of software supplier and is inescapable.

1.6. Objective elements in software licences

By objective elements, we mean those elements of the user licence relating to its object: the content and scope of the user rights. What rights are granted by the rightsholder to the user with respect to the software and subject to

what limitations? As we have already had occasion to note, the rights and obligations of the parties with respect to software vary substantially according to the licence.

In this section, we look at the term and price of a licence, and then the different rights that are granted.

1.6.1. Term

Software licences should establish the duration of the licence grant, i.e., its term. In principle, unless either of the parties were to breach their obligations under the user licence, it should remain effective during the established term. Generally speaking, licences are granted for a fixed term, an indefinite term, or sometimes do not provide anything in respect of term.

- Fixed-term licences. In fixed term licences[4], a specific period is established for the use of the software; n months, n years, etc. At the end of the term, if the licence does not say otherwise, it expires and the user must discontinue use of the software.
  This does not prevent the parties from agreeing later to subscribe a new user licence for the same software. It is even quite possible that the licence itself establishes that, when the effective term lapses, the licence should be deemed tacitly or automatically extended for a new term and so on until one of the parties gives advanced notice of their intention not to extend it any further.
  In the case of demonstration or evaluation software (known as demos), the licence is also usually established for a fixed term. In this case, fixing the term is essential to accomplish the purpose sought with the distribution of this software: for the user to get to know, over a short period of time, its functionalities and, at the end of the period, they may decide whether or not to purchase a complete version of the program.

(4)Licences granted for a fixed term are commonly used for more specialised and complex software applications, aimed at companies that are normally bound by an accessory agreement, such as a consulting or maintenance agreement. In such case, the user tends to pay regular instalments to the supplier as a licence fee.

- Indefinite-term licences. In this case, the software licence agreement expressly establishes that the licence is granted for an indeterminate period, not being subject to any specific term. Users may use the software as long as they meet the terms and conditions of the licence.
  Notwithstanding, some indefinite-term software licences[5] establish clauses that allow one or both parties to end the licence whenever they desire, by giving advanced notice of termination. This can be considered abusive in certain circumstances (see above, in respect of consumers).

(5)Indefinite or indeterminate-term licences are more commonly used for mass market software, especially for consumers, where the user pays the price of the licence on one single occasion.

- Lack of express term of the licence. When a software licence agreement does not specify anything with respect to its term, the licence is not nec-

essarily granted for an indefinite term. In certain jurisdictions, like Spain, in these circumstances the licence is limited to a specific term (five years, in Spain). This is often contrary to the intention of the licensor – who should improve the drafting of the licence!

This is what happened with the GPLv2. Although there may have been arguments to extend the period of the licence, on the basis that limiting their effective term to five years could go against the obvious intentions of the parties and the purpose of the licence, GPLv3, along with other more recent licences such as the OSL 3.0 or EUPL 1.1, now establishes the "maximum duration of rights" as the term.

- Term in free software licences. First, it should noted that free software licences are and should be granted for indefinite terms – which sets the term to the maximum duration of copyright protection. Thereafter, a licence is no longer needed as the software is in the public domain.
  To establish a limited term of duration in a free software licence would imply adding a restriction to user rights (in this case, a temporal restriction), which would be contrary to the very essence of the free software licence: not to limit the use of the software by the user, but to guarantee the freedoms over its use. It is thus commonly accepted that free software licences remain effective in time as long as the user respects their conditions.

  GPLv3 the OSL 3.0 and other modern licences, have filled the existing void under the prior version, indicating, for instance, in Clause 2 of GPLv3, that the rights granted under such licence shall be deemed "granted for the term of copyright on the program" and that they are "irrevocable provided the stated conditions are met". Likewise, other licences, such as the Apache 2.0, expressly indicate that they are granted with a "perpetual" and "irrevocable" nature (clauses 2 and 3).

1.6.2. Price

Another essential element of a software licence (at least in a most non-free licences) is the price, the amount of money that the user is to pay for taking the licence grant.

In terms of payment modalities, the price may be paid on one single occasion (lump sum), e.g. upon acquiring the licence. This is typical of mass-marketed software licences. Otherwise, payments can be made in regular instalments: the user makes a regular (monthly, yearly, etc.) payment of an instalment[6] to the supplier. This is typical of software licences for more specialised and complex applications, directed to companies, licences established for a fixed term and regularly bound by an accessory consulting or maintenance agreement, for which the user also pays a fee.

(6)Payment in instalments is now common for "software as a service", whereby the user contracts a (pseudo) licence agreement to use software – often over the web – for a monthly or period payment. We say "pseudo" licence because in many cases the user never in fact exercises any of the copyright protected rights (reproduction, transformation, distribution) but "uses" the services of the software. Basically, the user is paying an access fee.

When the licence may be extended in time, a clause is often included for the review or updating of the rate payable by the user. It is not valid to agree that the review of the instalment should be left to the free will of the supplier, but must be obtained either by mutual agreement between the parties, or referencing an objective index or parameter, such as the "consumer price index".

Licences may be granted for free, without the user having to pay anything for the use of the software. We must bear in mind that we must not automatically identify "proprietary" non-free software with paid software, and "free software[7]" with cost-free software. In the English-speaking countries, this confusion has arisen due to the fact that "free", in addition to "without restraint", also means "cost free".

(7)Free software is nearly always free (gratis), but many non-free programs are also distributed free of cost: Microsoft® Internet Explorer, internet messenger clients such as Microsoft® Messenger, Skype, etc., software demos, shareware or drivers.

Price in free software

As regards free software, we know that the term free does not mean that the program is licensed by the software supplier free of cost, but that it is licensed to allow users to use, modify and distribute it freely.

In the case of the GNU-GPL, the supplier may choose to distribute the software free of charge or in exchange for a consideration (paragraph 5 of the Preamble, and Clause 4 of Version 3); economic compensation may also be required for providing certain warranties on the software, unless required by law to provide those same warranties. Other free software licences, such as the Apache 2.0, expressly state that the licence is granted free of charge: its clauses 2 and 3 state that the licence is granted royalty-free.

Nonetheless, although the free software supplier may be entitled to require economic compensation, it is most common that the software is distributed free of charge and that the price requested is minimal (the term "residual" price is used), solely to cover certain expenses, such as the making of the copy, its delivery on a physical medium, etc.

Should an economic benefit be sought with the free software (which is not always the case), the supplier would not obtain it as much by charging a price for the distribution of copies, but rather for rendering services for the software, such as updates, consultancy and the marketing of copies of software based on free software. And on the market there are solutions based on free software that, considering the user licence on the software and the relevant consulting and/or maintenance services, have a high price (see, "Red Hat" as an emblematic case, and many others).

The fact that the supplier of free software cannot base their economic benefit on the price of the copy seems obvious: if the users are allowed to distribute the software freely, the supplier loses exclusive control over the copy. It does not make sense to charge a high price for the copy when the users could in turn distribute – online or on CDs – as many copies as they wish.

1.6.3. Rights, prohibitions and limitations

In prior modules, we have seen that author's rights or copyright legislation grants a series of important exclusive rights to the author of the software or the derived owner of the exploitation rights: the right to reproduce, transform and distribute (including, for our purposes, publicly communicate) the software. They decide what to authorise, when and how.

Additionally, we know that the software licence is the legal instrument whereby the software rightsholder allows its use by third parties, the users. The user licence therefore has an essential content:

- On the one hand, it establishes the rights that the rightsholder grants the user to the software: what the user may do with the software.
- On the other, it also establishes certain prohibitions and limitations on user rights, which the user must respect: what the user may not do with the software, and the conditions applied to its use.

We refer to Module 2 on authors' rights as to the scope of the rights that are exclusive to rightsholders and thus potentially subject to licence conditions:

- Reproduction.
- Transformation.
- Distribution (including rental).
- Public communication.

Adobe® Photoshop®

If you obtained the software and any required serial number(s) from Adobe or one of its authorised licensees and as long as you comply with the terms of this agreement, Adobe grants you a non-exclusive licence to install and use the software in a manner consistent with its design and documentation and as further set forth below... General Use. You may install and use one copy of the software on up to the permitted number of your compatible computers as long as, when required by the software, you present a valid serial number for each copy.

Generally speaking, all rights that are not granted in a licence are reserved, i.e. not granted. To reinforce this, licenses often add specific prohibitions:

Adobe® Photoshop®

4.3 No Modifications. Except as permitted in Sections 2.7 or 16, you may not modify, adapt or translate the software.

4.4 No Reverse Engineering. You will not reverse engineer, decompile, disassemble or otherwise attempt to discover the source code of the software except to the extent you may be expressly permitted under applicable law to decompile only in order to achieve interoperability with the software.

4.5 No Unbundling. You may not unbundle the component parts of the software for use on different computers. You may not unbundle or repackage the software for distribution, transfer or resale.

4.6 No Transfer. YOU WILL NOT RENT, LEASE, SELL, SUBLICENSE, ASSIGN OR TRANSFER YOUR RIGHTS IN THE SOFTWARE, OR AUTHORISE ANY PORTION OF THE SOFTWARE TO BE COPIED ONTO ANOTHER INDIVIDUAL OR LEGAL ENTITY'S COMPUTER EXCEPT AS MAY BE EXPRESSLY PERMITTED HEREIN.

It is common for rightsholders to attach conditions on the exploitation of the software. Some of these are reasonable (payment of a price, maintaining copyright notices), others may seem unreasonable or just strange: e.g. licences that forbid the publication of the results of any benchmark or analysis of the software. While these conditions are often outside the realm of copyright protection scope, if the licence is deemed a valid and binding contract, these provisions will be seen as contractual obligations binding on the licensee.

The most well known and highly debated condition in the free software domain is Clause 2.b. of the GNU-GPL that contains part of the copyleft obligations:

"b) 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 licence."

Applicable legislation itself may provide that, in the absence of express terms to the contrary, software user licences are granted to the user on certain terms.

In Spain, for example, licences are, by default:

- Non exclusive. In other words, they do not grant the right to use the software to a single user, but to a number of them.
- Non-transferable. The user cannot convey the licence to third parties, which also implies a prohibition to sell, rent, grant sub-licences or give away their copy, except with express authorisation from the supplier.
- Solely to satisfy the needs of the user. Without express authorisation, the user may solely use the software strictly for their own personal use, not to provide services to third parties.

1.7. Warranties and liabilities

A very important detail in supplier-licensor and user-licensee relations is the determination of the legal consequences derived from an incident with the operation of the software, especially considering the relative instability of software (it is susceptible of malfunctioning, mis-configuration, etc.) and the material inconveniences and damages that a user may sustain as a result of software issues (especially, the companies and entities whose activity depends on the proper operation of their information systems).

Intellectual property laws do not cover this aspect, but merely regulate the exclusive rights to the software. Nonetheless, various norms apply in all countries: Contract Law, rules on warranties in other agreements (such as purchase, lease or service agreements, applicable by analogy to the software licence), norms protecting consumers, etc., could oblige suppliers to assume certain warranties and liabilities with respect to the user, without the possibility of their being eluded by the licence.

1.7.1. General considerations

Software licences tend to regulate the rights of the user – and the consequent obligations of the supplier – in case any incidents were to occur with the software: malfunctioning, miscellaneous defects or if it does not match the characteristics that the supplier boasts in respect thereof and which led the user to purchase the licence.

When any of these circumstances occurs, the user is prevented from using the software or from using it for the purposes that led to acquiring the licence. Should the user not be at fault for the incident, a principle of justice tells us that the supplier-licensor should assist the user and put an end to the incident: we would thus be referring to the supplier having to provide the user a **warranty** in terms of conformity and the continued operability of the software.

Warranties

Warranties are the commitments or obligations assumed by the supplier-licensor in favour of the user, with respect to the conditions (characteristics, services, correct operation) that must be met by the software subject to the licence. This means that if the software does not meet or at some point ceases to meet such conditions, the supplier-licensor must take the appropriate actions for the software to meet them. Specifically, it should be noted that warranties of conformity and proper operation, whereby the supplier is to guarantee to the user-licensee that the software conforms to its description and will work appropriately during the effective term of the licence.

But what is more, such an incident could have caused damages to the user. We should think especially of the software on which, in practice, the proper day to day activity of a company or professional user depends: one defect or malfunction could paralyse their activity, which would obviously imply damages. If the supplier is "at fault" and, therefore, responsible for the damages suffered by the user as a result of the incident with the software operation, this would require them to provide **indemnity**.

Supplementary content

It could be said that "liability" (an obligation to compensate) is one of the possible consequences of the breach of a warranty.

We have seen that software suppliers tend unilaterally to impose clauses of the licences on the users. Licensors are particularly interested in establishing limitations or disclaimers of warranties and liability with respect to the software.

Nonetheless, in certain cases a disclaimer or limitation clause is not legal. The same principle of justice to which we referred earlier tells us that it would be unfair and/or abusive for the licence to allow the supplier to disregard any incidents occurring with the software. It would be particularly unfair if the user has had no opportunity to negotiate the content of such clauses, but rather were imposed by the supplier-licensor; or when the user has paid a price for the licence.

This situation would be different with licences in which the user-licensee has had the opportunity to negotiate the content of the licence and a disclaimer of warranties and/or liability in favour of the supplier-licensor, in exchange for a balancing item in favour of the user (for instance, a reduction in price or a better warranty in exchange for less liability). This would be the case with specialised software usage licences, highly-priced and adapted to the needs of the user. In such case, the limitation or exoneration could indeed be considered fair, as it would be freely negotiated between two parties in equal or similar negotiating positions.

1.7.2. Warranties

User licences usually regulate which warranties are to be provided by the supplier, their term and how they will be fulfilled: i.e., how the supplier-licensor would assist the user in remedying the incident, by repairing the fault or defect, substituting the copy with another, or refunding the price to the consumer, cancelling the licence.

In any case, the clauses of licences providing warranties, including their possible limitations or exonerations, must respect a series of imperative norms that, in each country, establish the requirement to provide certain minimum warranties with respect to the software.

The minimum legal obligations (warranties) on software are generally:

- The warranty to remedy any hidden defects.
- Conformity with specifications or description.
- Correct operation.

In the law of English-speaking countries, these are often called:

- Satisfactory or merchantable warranty. The software must be legally marketable –not be something prohibited– and must be of satisfactory quality, considering various criteria (price, market, state of the art, etc.).

- Fit for a particular (stated) purpose. The software must be fit to accomplish a particular purpose, when the licensee acquired the licence based on the possibility of accomplishing such purpose and the supplier knew or could have known that the licensee wanted to acquire the licence precisely for such purpose.
- Along with these warranties, there is also mention of a warranty of title and non-infringement. This corresponds with the warranty of "ownership", to which we have referred above.

In continental European law: There are different legal classes and categories of warranties as regards those inherent in the law of English-speaking countries. Nonetheless, many software licences, even written in a national language and to apply in that country, refer to the typical warranties of the law of English-speaking countries. This makes the wording of such clauses tend to seem confusing. In any case, the content and scope of the warranties is similar in either case, as are the actions and remedies established in favour of the user to implement them: repair, substitution of the copy or return of the price, cancelling the licence.

1.7.3. Liability or indemnities

Liability consists of the duty of the supplier to indemnify the user for the damages sustained thereby as a result of an error, defect or malfunction of the software, or of its lack of suitability for the characteristics that could be expected thereof.

It may be the case that, by reason of an incident with the software, the user could sustain damages. In such case, it would not suffice for the user that the warranty should be honoured (that the supplier should repair the malfunction, provide a new copy or return the price paid). The user shall also seek to obtain compensation for the damages from the supplier to the extent that they are the responsibility of the supplier.

The example that comes to mind is a company that has to suspend its activities due to a failure in the operation of a computer application. In such case, if the company suffers losses (unrealised business, salaries paid to employees that cannot work, etc.), it may seek to demand indemnification by the supplier.

To consider the supplier to be liable for the damages, the fault or defect causing them must not have been fortuitous or the exclusive fault of the user itself, but rather must be attributable by some means to the supplier-licensor:

- Either for what is legally known as **wilful misconduct**: when the supplier was aware of the existence of the malfunction or defect in the software that caused the damages to the user.
- Or for fault or **negligence**: when the supplier was unaware of the existence of the malfunction or defect but should have known, had they performed their duties of programming or maintenance of the software with the degree of diligence expected from any programmer.

In addition, liability can arise for a variety of types of damages, direct or indirect. As with warranties, software licences tend to reference the types of damage contained in the law of English-speaking countries. Generally speaking:

- **Direct or incidental damages**: those that are the direct result of the incident (for example, again, the loss of information or the expenses for the reconstruction of the lost information).
- **Indirect or consequential damages**: those indirectly derived from the incident, whether the parties knew or should have known that they could

have been sustained in the event that such incident were to occur (for instance, loss of reputation with clients).

- **Lost profits**: There are certain damages, such as the loss of profits, which would in principle be included as indirect damages. Nonetheless, on occasions, the criteria of the British and United States courts have varied, including them sometimes as direct damages. Therefore, licences tend to cite loss of profits separately.

As regards Spanish law, for example, the following are defined:

- Consequential damages: the value of the various equity and moral losses directly sustained by the user as a result of the incident (for instance, if a software malfunction causes a loss of information, the value of such lost information; or damage to the image that an entrepreneur user sustains with respect to clients), and the expenses incurred to remedy such incident.
- Lost profits: profits not obtained by reason of the incident (for instance, the income that the entrepreneur user does not receive during the time in which their activity is suspended due to the software malfunction).

1.7.4. Limitations and exclusions of warranties and liabilities

Suppliers tend to include warranty and liability limitation and disclaimer clauses in software licences. Although the principles of law generally allow for contractual freedom (in determining the contracting conditions) the legal effectiveness of such clauses is questionable.

Warranty disclaimers

Software suppliers-licensors seek to avoid certain warranties that they should provide the user or to shorten the term for which they should be provided, imposing warranty limitations or disclaimers for such reason in the text of usage licences. Should the licensee consumer have agreed to the licence agreement without negotiation, under the laws for the consumer and user protection these limitations could be declared null and void and be left out of the agreement for being abusive. Therefore, the minimum legal warranties that we just described cannot be limited in such circumstances.

Nonetheless, if the licensee is not a consumer but a business or professional, the supplier may be entitled to restrict its warranties and liabilities in the licence. Nonetheless, they cannot shirk any incidents occurring to the software quite so easily and the licensee could resort to the analogous application of the (Civil or Commercial Code) rules providing for a warranty of repair of hidden defects, applying the principle of good faith in agreements and others, to demand that the supplier should ensure that the software should remain in perfect state of operation. The validity or absence thereof in limitations of warranties must be determined on a case-by-case basis.

Liability limitation and disclaimer

Additionally, software licences typically include clauses of disclaimer of the liability of the supplier with respect to damages sustained by the user due to incidents with the software. Or, if indemnification is due to the user for damages, they limit the possible indemnification of the supplier: for instance, it is common to limit it to an amount equivalent to the price that the user has paid for the licence (along with the payments made for maintenance services).

In many cases, the validity of these liability disclaimers or limitations is dubious at best. What can be said with certainty is that the liability disclaimer or limitation shall not be valid when:

- The liability derives from wilful misconduct: wilful misconduct not only occurs when the supplier causes the damage knowingly (which would not be very normal), but also when the supplier knows of the existence of an incident that could cause certain damage to the user and does nothing to prevent the damages.

  Gross negligence. In countries other than Spain, just as wilful misconduct, liability for gross negligence cannot be subject to limitation: when the incident is attributable to the supplier, due to a lack of the diligence expected from any supplier.

- Liability for damages consisting of the death or corporeal damages to people: in principle, we find it hard to think of a program whose malfunctions or errors could cause such damages, except in certain specific cases such as the software of a medical device, applications used by air traffic controllers, etc.

- When the licensee suffering the damages is a consumer: under the provisions of the Consumer Protection Law, consumer users are entitled to compensation for the damages sustained as a result of the malfunctions or unsuitability of the software, unless these are their own exclusive fault. Therefore, if the licensee is a consumer, the supplier cannot validly reduce the liability limitation to a maximum amount, as such a clause would automatically be null and void for infringing upon the law and for being abusive. Nonetheless, when the user-licensee is a business or a professional, the liability limitation to a maximum amount is valid in principle, as the legislation allows the parties to agree on this issue.

  This is important as suppliers are especially interested in limiting their liabilities with respect to licensees that are businesses or professionals, as software malfunction could imply for them damages of much greater importance –at least in economic terms – than with a consumer.

  Notwithstanding the foregoing, even if the user-licensee were a business or a professional, it would be necessary to study each specific case to determine whether the limitation of liability could be especially unfair and abusive. In such case, the limitation could be declared null and void, if it were considered that the limitation of liability is so disproportionately abusive that:

- in practice, it implies making the supplier totally irresponsible for its own obligations; and/or
- it breaches the principle of good faith in agreements.

In the UK, case law generally has established that when the licensee is a business or professional, the supplier-licensor may limit its responsibility under the licence when doing so is not "unreasonable". To determine whether liability disclaimer or limitation is reasonable, the courts take into consideration various circumstances, known as the reasonableness test:

- Whether there has been a true process of negotiation of contractual clauses, particularly those relating to warranties and liabilities. Or whether it was the opposite and the supplier imposed the content on the licensee.
- Whether the licensee knew of the existence and scope of the limitation clause, whether they were under advice from counsel to inform them in that sense before signing the licence.
- Whether the limitation or disclaimer clause was accepted by the licensee in exchange for something in their favour (for instance, a price reduction).

1.7.5. Warranties and liability in free software licences

It is said, and it is true to a great extent, that free software licences are granted with no warranty whatsoever for the user and that no sort of liability is assumed. This is debatable, especially in the legal framework of European and Member State laws, especially those seeking full exoneration of liability.

GPLv3

"there is no warranty for the program, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair or correction".

Governing law and the facts of the case will ultimately determine whether the supplier of free software should provide a warranty or whether they incur any liability with respect to the user-licensee.

The validity of such absence of warranties could be upheld when the free software is distributed free of charge and the limitation of liability is subject to "the extent permitted by applicable law". Indeed, it could be said that the distribution of free software may be equated to a gift. And the rules governing gifts generally do not compel the giver (in this case, the supplier) to provide warranties regarding the gift (in this case, the right to use the software) with respect to hidden defects or to insure its proper operation. If the gifted item proves defective, the giver usually has no obligation, in principle, to repair it or substitute it with another. The liability would be different in the indemnification of damages sustained based on a malfunctioning or defect.

In any case, we must take into consideration that not all free software is entirely "free". When the licence for free software is granted accompanied with the supply of additional services, such as maintenance or update services (the case of "Red Hat" for instance), the supplier charges to provide such services. It

must therefore comply with obligations with respect to the proper rendering of the services (proper choice of a free software solution, good adaptation and implementation for the user, etc.).

There is also the question of whether it is the licensor himself who should provide the warranties (certainly as to title), or the person who supplied the software (probably as to quality and fitness).

With respect to limitations of liability, the question of the validity of the clauses is more doubtful:

GPlv3, Cl.16

"In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who modifies and/or conveys the program as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages."

Regarding this liability disclaimer, there would not seem to be many circumstances –as regards free software– that would allow the supplier to disavow the applicable legal system, which prohibits absolute liability disclaimers. Furthermore, as we have already seen, there can be no disclaimer or limitation of liability, when derived from wilful misconduct or if the licensee were a consumer. In other words, this clause would be ineffective with respect to a licensee consumer.

The Free Software Foundation itself and other entities developing free software projects are aware that some warranty and liability disclaimer clauses, in absolute terms (as is), have validity issues in many countries. In this sense, warranty and liability disclaimers tend to include a typical mention that such exonerations are valid "except as required by applicable law" or "to the extent permitted by law".

Version 3 of the GNU-GPL has complemented the qualification with its clause 17, destined for interpreting the warranty and liability disclaimers established in Clauses 15 and 16: "If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the program, unless a warranty or assumption of liability accompanies a copy of the program in return for a fee".

Beyond the legal effectiveness of the warranty and liability disclaimer clauses, free software licences tend to provide that a licensor may choose to voluntarily provide some type of warranty for the software or assume some degree of liability, in principle in exchange for an economic consideration (for instance, under the framework of the rendering of maintenance services).

An important aspect is that, in these cases, the licensor of the free software assuming warranty or liability commitments with the users, does so personally and does not thereby bind prior licensors of that same software, from whom the free licence was acquired.

Other free software licences establish similar provisions, such as the Apache 2.0 licence (clause 9), for instance.

1.8. Jurisdiction and applicable law

Many software licences expressly establish a clause regarding competent jurisdiction and governing law, which is of great importance in case of conflicts between the parties leading to litigation. These clauses are especially relevant, obviously, when the supplier and licensee reside in different countries.

Under such agreement, the licence determines:

- **Jurisdiction**: the courts of the country (region, city, etc.) that shall have competent jurisdiction to resolve upon any litigation between the parties derived from the licence. Therefore, if one party wants to claim something from the other, it must do so before the courts agreed to have competent jurisdiction.
- **Applicable law**. The law (laws, regulations, etc.) of the country should govern in applying and interpreting the clauses of the licences. In case of litigation, the court or arbitrator designated as having competent jurisdiction must resolve upon the matter in accordance with the law agreed by the parties to be applicable.

Should the parties not have expressly agreed upon the competent jurisdiction and/or law applicable to the licence, it would be necessary to abide by what is determined in the norms on "Conflict of Laws[8]" of each country.

(8)The area of Conflict of Laws, or Private International Law, is extraordinarily complicated, and even more so in the realm of intangible transfers, downloads, content management systems and worldwide audiences, web-services and software as a service.

For the purposes of this section, it is sufficient to know that, in principle, agreements reached between business parties establishing competent jurisdiction and the law applicable to the agreement in the licence are valid.

An exception exists, once again, when the user-licensee is a consumer. In such case, the user may sue the licensor both in the courts corresponding to the domicile of the licensor and those of its own domicile (which would undoubtedly be more convenient and economical for the user). On the other hand, if the licensor seeks to sue the user, it may solely do so before the courts of the domicile of the consumer.

Even if an applicable law other than that of the country of residence of the user is agreed, the user may regardless seek the application of the consumer protection laws of their country of residence.

Consider a software licence in which the supplier is from the United States and the user is a French consumer. If, for instance, the licence establishes that the competent courts are those of the United States and the applicable law is that of the Federal Law of the United States and the Law of the State of California:

- The user could also sue the supplier in France and the French courts would be deemed to have competent jurisdiction.

- The user could invoke the application of the Consumer Protection Law and other norms protecting consumers in French.

However, for licensee consumers to be able to benefit from these norms that protect their interests, they must have contracted the licence with the supplier, who must have engaged in any sort of commercial activity specifically directed to the country of residence of the user (advertising, opening of a store, etc.).

This is important considering the numerous licences contracted over the internet, particularly on the websites of software suppliers. In principle, in case of litigation for a software licence acquired over the internet, the user consumer could only benefit from the aforesaid protective norms if the website is directed specifically to their country of residence (either in conjunction with other countries or on its own).

Continuing with the last example, let's assume that the licence corresponds to software downloaded off the internet. It would be necessary to verify whether the website of the United States supplier is by any means specifically directed to France (for instance, through such expressive signs as having a section in French, showing a price in Euros or indicating technical service or a branch of the supplier in France). It would be in such a scenario that the licensee consumer could sue the supplier in France, in the event that a dispute was to arise between them, and require the application of French consumer protection regulations.

2. Software contracts

Most software is not created by an independent programmer for his/her own exclusive use or that of few people. It is created by companies that precisely develop and distribute software for third parties that they trust has uses and applications that will satisfy the expectations and needs of determined users. In this section, we comment on the licensing issues relevant to certain business models.

A software licence adapts to all types of software, both "standard" (for an indeterminate amount of users), and "customised" (commissioned by a client), and "parameterised" software mentioned below. In each case, the original software owner (and the owner of the modifications or parameterisations) must grant user rights to the user. The difference lies in that for "standard software", a "standard" user licence is used (the EULA of Microsoft Windows or the GPL), while for "customised" or "parameterised" software there tends to be negotiation between the parties in terms of the legal user conditions.

2.1. Standard mass market software

The purpose of companies developing mass market or "standard" software (usually without adaptations to the particular needs of the user) is to distribute software among the largest number of users possible to obtain its utmost dissemination and, why not, the highest economic benefits. It is said, for that reason, that this software is "mass" traded or distributed.

Often standard software serves to cover more than the needs of the users acquiring it – think of all the macro functions of word processor software that the everyday user doesn't even know about, let alone use. Users, in turn, find it much more economical to purchase a standard software licence than to commission a programmer to make them a "customised" word processor, for instance.

In the case of "traditional" non-free software, the income is greater the more copies of the program are sold. Licences are drafted to prevent reproduction and redistribution of the software, which would eliminate revenue, and also prevent modification, which could give rise to malfunctioning and difficulties to provide user support and maintenance services (patches, etc.).

Example

E.g.: Symantec Product License agreements.

2.2. "Bespoke" software

When a programmer or a programming company creates (unique) new software on commission from a client, adapted to its needs, and which must satisfy the instructions and exclusive needs of such client, the result is called bespoke software.

The contractual relation between the two parties is governed by a "software development agreement", which governs specifications, delivery, acceptance, guarantees, price, etc.

One of the most important clauses of a software development agreement is the ownership or "title" clause and copyright licence, determining who is to own the software created by the programmer and what are the exploitation rights.

- Title may be attributed to the client ("*work for hire*" model, in the USA). In this case, the programmer (author of the software) assigns the exploitation rights to the client.
- Title may remain with the developer. In this case, for the client to be able to use the software, the development agreement provides that the programmer grants the client a user licence.

The developer may also maintain control over certain parts of the bespoke software that are used "generically" in their developments (licensing it to the client) and assign the exploitation rights to the part that is truly "customised" for the client.

2.3. Customised software

Additionally, in many enterprise situations, adaptations may be made to a standard application, such as Enterprise Resource Planning software (ERP), Document Management Systems (DMS) or Content Management Systems (CMS), according to the particular needs of each client. The adaptations are often called customisations or parameter changes ("parameterisations", also known as "extensions"). This is more in the line of a service agreement, with the client taking a licence from the manufacturer of the standard application, and hiring the services of the developer for the customisation and deployment.

In these circumstances, although the application software may be non-free, the fact that the solution incorporates adaptations conforming to the needs of the user may also imply that the user-licensee may have a limited right to modify the software and access the source code; specifically, to create, modify or remove the customisations or parameterisations.

In the cases of both bespoke and adapted software, the project may be described as "turnkey", whereby everything is supplied to the client in an immediately working condition. In these agreements, not only is a user right granted to the user for the software, but also a warranty or specific result in their favour is expressly agreed, i.e., the software satisfies the specific needs of the user-licensee. A "turnkey" licence may, in principle, apply to either non-free or free software.

2.4. "Mass" contracting and general conditions

As seen before, the software licence is an agreement between two parties. Nonetheless, almost always one of them (the software supplier or licensor) unilaterally establishes the terms and conditions of the licence. In such case, the user cannot negotiate the licence conditions with the licensor, but must merely accept or reject them. This is a logical consequence when dealing with standard software, destined for "mass" distribution, whether or not it is intended to obtain an economic benefit.

The software supplier clearly cannot and does not want to negotiate the terms of the licence with each of the hundreds or thousands of users. Rather, on the contrary, the supplier wants all of them to use the software in accordance with the same conditions imposed thereby. Clearly, if the user does not accept the conditions, they do not acquire the licence and, accordingly, cannot use the software.

When a software supplier imposes upon all users the same terms and conditions of the software licence, which they may only accept (if they wish to use the software) or reject, we are dealing with an "adhesion agreement" and it is said to be based on general conditions.

> On some occasions, licences also have particular conditions, applied solely to a specific contractual relation: for instance, if the licence contains any clause regarding the adaptation (customisation) or "parameterisation" to the specific needs that a particular user has indicated to the supplier.

The use of general conditions is subject to the meeting of certain legal requirements. In the European Union, laws apply regarding the general contract conditions seeking to protect the position of the contracting party "acceding" to the conditions in the event of abuses by the entrepreneur or professional imposing them.

> The main requirements are usually that:
>
> - The general conditions must be drafted precisely, clearly and simply.
> - The acceding party must have been allowed to go over them before accepting the agreement.
> - The party imposing them cannot benefit from an unclear or ambiguous wording: in case of doubt in terms of its interpretation, the clause shall be construed in a sense favouring the acceding party.
> - If a general condition has a content that is incompatible with a particular condition, the particular condition shall prevail.

- Furthermore, if the acceding party is a consumer, as we have seen, some clauses are considered abusive and cannot be imposed, as they are considered unfair and disproportionately unfavourable for the consumer.

These will in fact also apply to free software licences, though it would be difficult in any circumstances to argue that the terms of the licence are abusive, given the extent of the rights that are granted, the few limitations, and the non-cost free nature of the software, in most circumstances. What could fall under scrutiny are the limitations on warranties and liability, which are likely to fall foul of consumer protection based legislation.

2.5. Agreements ancillary to the software licence

Along with software licences of a certain complexity, directed to companies, there can be certain additional service agreements, which we shall refer to as "ancillary agreements" to the software licence, as their existence is dependent on the software licence to which they are associated.

These agreements may be contained in a document separate from the user licence, but may also form part of the licence agreement, whether incorporated among its clauses or as an attachment thereto. Among the most noteworthy of such ancillary agreements are maintenance agreements and consulting and training agreements (which are sometimes combined with the former).

Maintenance agreements

Software operation is relatively unstable and its possible malfunctions are not easy to repair, especially if the user does not have the source code. It also becomes obsolete quite quickly. Therefore, once the warranty period offered by the supplier for the software with the licence has ended, it may be essential for the user to maintain the software, especially the software of some complexity, destined for businesses and professionals. For the software supplier, providing the maintenance service will imply a complementary, and even a quite important, source of revenues. Providing a maintenance service also allows improving the software and repairing any faults advised by the users.

Through the maintenance agreement, the service provider undertakes to the user to maintain the proper operation of the software and/or to provide successive new versions, in exchange for a maintenance fee (annual, quarterly, monthly, etc.).

> In the case of non-free software, the maintenance service may often solely be provided by the software supplier or someone authorised by the supplier: they are the only ones with the source code and the only ones with the right to modify the licensed software. Users may be "captive" of the software manufacturer or service provider.

With free software, the business model may be based on providing such services as maintenance, but in this case, for a different reason. It is not a matter of more income for the software owner (the income for granting the licence is nil or minimal), but as there are no exclusive rights to the software, the

maintenance services are provided in free competition. Software integrators and consultants may compete among each either to provide a better, cheaper or more reliable maintenance service.

> With FOSS, anyone with the appropriate expertise could provide this type of service, enabling users to shop around and change support provider. Many professional or enterprise free software projects, such as Red Hat, Alfresco, Pentaho, etc., use this type of agreement as a significant revenue stream in relation to the licensing of their free software.

Service modes: there are typically three forms of maintenance service. Many agreements establish several or all of these modalities:

- Corrective maintenance: technical assistance to correct the errors or malfunctions in the operation of the software.
- Preventive maintenance: technical assistance through regular reviews, for instance, to avoid errors from occurring during operation.
- Development or update maintenance: consisting of providing the user with the improvements or new versions successively launched to market by the supplier.

Consulting and training agreements

Software suppliers often provide a service to users that consist of attending to inquiries relating to the selection, integration, installation and operation of the software. This is distinguished from the maintenance agreement in that, in this case, the purpose is not to avoid or correct issues with the software, but to create a solution for the needs of the user and to resolve any doubts relating to its operation for the user.

Being a service that users need, especially at the beginning of their use of the software when they are still not well acquainted with it, this service could possibly be included with the licence as ancillary to the subsequent installation of the program.

As part of the consultancy agreement, a possible variation lies in training: when the technicians of the supplier teach the employees of the user how to operate the software or teach courses on its use.

As for maintenance agreements, in the case of free software, the consulting and training service may be rendered in circumstances of free competition by any computer services company, as access to the source code of the program is open and anyone can gain sufficient skills to install, develop and train clients on the software.

3. Free software and free content

We have seen in the introduction to this module that the free software and content movement (including for these purposes, the open source and the free content movement) has positioned itself as the defendant of access to and dissemination of certain forms of culture and knowledge in an increasingly restrictive society, where IP laws are used to control the exploitation of works to an ever greater extent, in the face of technological change.

The focus of this section therefore is to understand the basic concepts of the free software and content movement, before looking in more detail at free software and content licences.

3.1. Free software

Although a precedent exists at the University of Berkeley and in the BSD licence that we shall comment on below, for many the founders of the free software movement, round 1983-1984, were Richard Stallman and the Free Software Foundation. Richard Stallman, who at the time was employed at the MIT AI Lab, abandoned his work to undertake the GNU ('GNU is Not Unix') project and founded the Free Software Foundation to obtain funds and a more formal structure for the development and protection of free software.

> Richard Stallman established the ethical foundation for free software in such documents as "The GNU Manifesto" and "Why Software Should Not Have Owners". Since the beginning of the GNU project, Richard Stallman was concerned with the liberties that would be available to the users of the software created. He is interested in that, not only those receiving the programs directly from the GNU project, but also those receiving them following any number of redistributions, could continue to enjoy the same rights (modification, redistribution, etc.).

The basic tenets of the free software movement is the need to ensure that users of software have significant freedom (in legal terms, rights) to exploit software, understand it, learn from and it share it with third parties.

The Free Software Foundation established a core definition for free software: software under a licence that allows and guarantees the exercise of the following four freedoms to the users:

- The freedom to run and use the software for any purpose (freedom 0).
- The freedom to study the program and adapt it to your needs (freedom 1).
- The freedom to distribute copies (freedom 2).
- The freedom to modify the program and release the modifications to the public (freedom 3).

> See online at the GNU site. The importance of this definition is twofold. On the one hand, the free and open source community is in agreement with it and respects it; even thought different parts of the community may have differing philosophies or views. On

the other hand, from a legal perspective, it is a unifying tool for the analysis of free software licences: it separates what is free from what is not.

To enjoy such freedoms, especially 1 and 3, the user must have access to the source code of the program. Free software licences indeed contain a commitment by the supplier-licensor to provide the source code to users or, at least, to make it available to them. Below we shall briefly analyse how the users are granted the rights of use, copy, modification and distribution in free software licences.

In English, the word free has two meanings: 'unencumbered' and 'without charge'. It should therefore be clarified at this time that the use of the term "free" in relation to software does not imply that the owner or provider of the software provides or distributes it free of charge (although they may). The term free refers to the software being distributed under a licence that allows users to use it freely. As regards the economic consideration for the distribution of free software, we shall see that most licences allow the distributor to use the price of their choice.

The BSD free software licences allows code to be privatised or "closed" and, therefore, its sale as a commercial product. The General Public License (GPL) explicitly allows charging for distribution (clause 1). The price is solely limited by the rule of market: as the user could subsequently publish the source code on the internet or by any means distribute it free of charge, any third party could obtain a copy without paying.

Why not public domain?

Without doubt, the simplest way to make a program free is to make it an object of public domain, with no rights reserved. This allows the creator to share the program and its improvements with the entire world with no restrictions. But this solution will allow third parties to use the software in a manner that may go against its original philosophy, making it non-free or closed software. To avoid such a possibility, the FSF created the concept of *copyleft* and protected the GNU and software against future intermediaries that could attempt to restrict the freedom of the users to redistribute and change it.

Additionally, as we have studied, in continental systems, moral rights are inalienable, which means that it is also impossible to voluntarily place software under public domain, waiving the moral rights thereupon. Nonetheless, even in English-speaking countries (where the figure of "moral rights" does not exist as applied to software and software may indeed be placed voluntarily under public domain), which is where free software licences originate from, it has also been sought to clearly provide that the original author of the free software does not waive their status as such. Therefore, it is common among the various modalities of free licences to maintain a notice of authorship.

3.2. Copyleft

Thus, if free software rights are granted unconditionally, a user would be free to incorporate the software and any work resulting from using the software into a proprietary or non-free program. According to the supporters of this movement, for the free software philosophy to be truly effective, derivative versions must also be free. The goal of "*copyleft*" is to establish a licensing framework whereby the essential freedoms are granted but free software may not

be transformed into non-free or closed software. Copyleft guarantees that all derivative work, based on the free software distributed with copyleft, shall be available under the same free terms.

Copyleft may thus be defined as a manner of licensing rights in a work with the particular condition obliging redistribution of the work, and any derivative work, to be on terms that maintain the freedoms of use, modification and distribution for all future users and licensees: no further restrictions may be added.

Supplementary content

In this sense, the code and essential freedoms defined by the free software are legally inseparable. The freedoms are guaranteed for anyone having a copy and become inalienable rights.

To accomplish this objective, R. Stallman wrote the General Public License (GPL) as a foundation to guarantee the freedom of all free software users at all times. Thus the GPL goes beyond guaranteeing the four basic freedoms, and includes terms compelling the use of the same licence (the GPL) when redistributing both the original software and any work derived from it (and potentially any other work including it) and offering access to the source code. In other words,

- Redistributors are not allowed to add additional restrictions to the licence (other than those of the original GPL).
- In general, they must accompany any binary code with the relevant source code.

This mechanism is also used in other licences, including the Lesser GPL (LGPL), the Mozilla Public License (MPL), the Common Public License (CPL) and the Open Source License (OSL), discussed below. Most of these are characterised by a "weak" copyleft, as they solely affect the original software (and the derivative works), and not the works using or containing the software with such licences.

However, the GPL is important not only because it is the most used licence in the free software world (accounting for 70% of the free projects on Sourceforge) or because it is the precursor of many other current free licences (not all of them, though, as the BSD predates it), but because the principle of freedom of the FSF has been the basis and one of the most outstanding elements of the free movement.

On the basis of this copyleft condition, the pool of software subject to copyleft available to all may only increase as new applications are created by developers based on the software distributed under a copyleft licence.

The term copyleft is based on a play on words: copyleft uses copyright laws and legal framework, which is basically restrictive, but turns them around to serve the opposite of their usual purpose. Rather than being a means of keeping software private or undisclosed, it becomes a means of keeping it free. The developers of non-free software use copyrights to restrict the freedom of the users and to restrict its free reproduction; the GNU movement uses the reserved rights to guarantee their freedom and that is why they reversed the name.

This copyleft concept has had phenomenal success in the sector and the FSF is the institution *par excellence* defending the values and ethics of the free software movement. In 2006-2007, the FSF reconsidered the GPL, then in its 2nd version, in the light of the technological and legal changes, and presented to the community a draft of a new version of the GPL (GPLv3). The process of drafting of the new licence, up until its final approval in June 2007, was

complicated and protracted and stakeholders were as varied as multinational companies, the academic sector, the individual developers and the public administration.

The specific implementation of copyleft used for most GNU software is the GNU General Public License, or GNU GPL for short, and the GFDL for GNU manuals (using a copyleft adapted to documents). There is also the GNU Lesser General Public License (GNU LGPL), which is applied to some GNU libraries. Other copyleft licences (of various types) include the Open Source License, the Common Public License, the Mozilla Public License, and others that we will comment on below.

We stress that the Free Software movement by the FSF is essentially a philosophical, ethical and political movement. It is not a technological organisation or free software project (that would be the GNU project, the project closest to the FSF).

It should also be noted that copyleft does not affect the rights of use of the original licensee (an end-user, for instance), but restricts the freedoms relating to the subsequent distribution of the copyleft software with or without modifications (or closely incorporated in other applications). To understand this allows understanding why a copyleft clause does not affect the commercial use of applications subject to copyleft at private or public organisations, as such organisations are normally end users.

It is also necessary to stress that the legal impact of the copyleft clauses has led to great concern in the software world in general. It has especially been feared that the interrelation or incorporation of code with the GPL in other programs could affect the use or distribution of the resulting application or that the use of software with the GPL (for instance GNU/Linux) could prevent the use of other non-free applications. These doubts, which are often myths, are addressed below.

3.3. The Open Source Initiative and open source software

In 1998, there was a certain conceptual difference of opinions in the free software movement that in truth merely brought to a head the division that had been seen since the early nineties. This division gave way to the creation of the Open Source Initiative (OSI), which established the open source definition to determine whether a licence was "open source" or not (OSD).

For some, the term open source is a modality of free software; for others, it is a general term that encompasses all free software and, finally, for others, it is a dangerous departure from the original concepts of free software to achieve enhanced commercialisation.

The Open Source project was born from a strategic meeting held in February 1998 in Palo Alto, California, to react to the plan hatched by Netscape Inc. to release the source code for its browser, the Netscape Navigator. Among the present were Eric Raymond, Bruce Perens (then leader of the Debian group), John "Maddog" Hall (from Linux International) and Sam Ockman (representative of the group of Linux users from Silicon Valley).

See "Open Sources: Voices from the Open Source Revolution" (O'Reilly, 1999).

The OSI seeks to reconcile the freedoms of free software (in general) with the commercial needs of the companies involved in the creation, distribution and use of free software. By doing so, open code software maintains the fundamental freedoms of the free movement (reproduction, transformation, distribution, access to source code), but not the name "free software". It is replaced with "*open source software*", as the OSI considers that the excessive emphasis made by the FSF on moral or ethical reasons for the freedom of software could cause negative reactions on the business mentality and that it is more beneficial to promote free software on its technical merits.

> On the other hand, note that the FSF does not agree with the use of the term open source to refer to free software, precisely as it makes it lose the ethical dimension referring to freedom. See "Why Open Source misses the point of Free Software", R Stallman.

As a result of this initiative, the Open Source Definition was established. The OSD was designed to establish an open and understandable statement of the principles of the free software movement and a system for the classification and "certification" of the variety of free licences in existence. It is argued that, by establishing standards in this manner, the definition allows developers, users, commercial organisations and the public administration to better understand the free software movement and better respect its principles.

> It should be noted that open licences are free licences and vice versa. The difference between the OSI and the FSF (as institutions) is in their perspective (marketing, underlying philosophy, etc.) and not their principles in relation to licensing, which are shared by the two entities. In truth, the differences are not legal, but of position –the OSI stressing more the need to access the source code and the FSF placing more importance on the ethics or philosophy of "freedom". It is clear that the GPLv2 and the GPLv3 are "open" licences, conforming to the OSD: they are OSI certified.

The Open Source Definition (OSD)

The OSD was born from the *Debian Free Software Guidelines* (DFSG), revised in 1998, basically to eliminate references to Debian. The definition of open source software in the DFSG was indeed broad enough to include such licences as the BSD, the GPL and its sister the LGPL, and such others as the MIT/X and the Apache. Its requirements were therefore adopted by the OSI as general guidelines to be met by all open licences.

The definition of the OSI stresses the four fundamental elements of the free software movement, expressed in the four freedoms listed by the FSF. Additionally, availability and access to the source code is fundamental: the word *open* could be better translated by *'available', 'visible'* or *'readable'* and we could speak of *available* source code software licences.

The Open Source Definition

Introduction

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

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

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicised means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

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

4. Integrity of The Author's Source Code

The licence may restrict source-code from being distributed in modified form only if the licence allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The licence must explicitly permit distribution of software built from modified source code. The licence may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups

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

6. No Discrimination Against Fields of Endeavor

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

7. Distribution of Licence

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

8. Licence Must Not Be Specific to a Product

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

9. Licence Must Not Restrict Other Software

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

10. Licence Must Be Technology-Neutral

No provision of the licence may be predicated on any individual technology or style of interface.

Example

An interesting example of the OSD application is seen in the case of KDE, Qt and Troll tech. KDE is a desktop graphic interface for Linux and depends on graphic libraries called Qt, owned by Troll Tech. Nonetheless, the Qt licence did not conform to the OSD, as a special licence was required to incorporate such libraries in applications that were not X Windows System. (Qt obtained income for the assignment of licences to Microsoft and Apple). Therefore, the free application KDE incorporated elements that were considered not to be free. Under pressure from the free community and the OSI in particular, Troll Tech agreed, initially, to create a special licence to release the Qt code in the event of the merger or bankruptcy of the company. Later, at the beginning of the development of GNOME, an open product competing directly with KDE, and with the creation of free libraries similar to Qt (such as Harmony), Troll Tech modified its licence to conform to the OSD.

It was argued that by establishing standards in this manner, the definition would allow developers, users, commercial organisations and the public administration to better understand the free software movement, enhance respect for its principles and, why not, find new business models to guarantee their future.

The OSI has further prepared a certification mark, the OSI Certified, which is a clear means of indicating that a licence complies with the OSD. The mark also serves to distinguish the term general open source, which has not had a sufficiently-defined use to guarantee such conformity.

3.4. Free software licences

Both the FSF and the OSI implement their philosophy and strategy through a work tool of legal nature: a free software licence. The FSF, by publishing the GPL and LGPL, and now the Affero GPL (AGPL, specifically designed for software distributed as a service or the offering of remote software services). The OSI, due to the cataloguing and classification of the various free licences used more or less by the community, published on its website: www.opensource.org.

As seen earlier, the difference between free software and non-free software lies in the rights and obligations specified in the licence. Those granted under free software licences offer a broad freedom to exploit the software, in terms of its use, modification and distribution, and tend to be directly opposite to those granted and reserved by a non-free software licence ("non-free licence").

We should recall that the four freedoms correspond to exclusive exploitation rights reserved to the owners of author's rights by applicable law:

- Freedom 0: the use right (not an exclusive right, but the free licence allows unrestricted and indiscriminate use).
- Freedom 1: the right of modification.
- Freedom 2: the rights of reproduction and distribution.
- Freedom 3: the rights of transformation and distribution of derived works.

All free software licences must therefore license these rights to users.

> For instance, the MIT licence establishes that "permission is hereby granted [...] to any person obtaining a copy of this software [...] to deal in the software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sub-licence, and/or sell copies of the software [...]", while the BSD licence provides that "redistribution and use in source and binary forms, with or without modification, are permitted [...]".

It is important to understand that not all free software licences are the same. The range of possibilities span from some minimum requirements (e.g. the BSD and MIT licence), solely requiring the maintaining of the copyright notice and warranty and liability disclaimers, up to the "maximum" (in certain sense) of the copyleft clause of the GPL, requiring the user to distribute any modifications and derivative work under the same GPL.

> As a result of the obligation to freely distribute any modified or derivative work, it has been said that "the GPL is not as free" as other free licences. The FSF rejects such classification arguing that, quite the opposite, the GPL is freer, as it guarantees greater freedom for the end user. Consider the following:
>
> - The BSD, for instance, grants the developers more freedom, as they may incorporate and distribute implementations of "BSD codes" under both types of licences, free and non-free.
> - The GPL gives the end users greater freedom as they always receive applications with open source code and a free licence.

Briefly, free licences may be classified into three categories: permissive licences, free licences with strong copyleft and free licences with weak copyleft.

Copyleft licences

GPLv2, GPLv3, AferroGPLv3, Sleepycat

- **Permissive (or "academic") free licences.** The Berkeley Software Distribution (BSD) licence is perhaps the simplest version of all free licences, and is also the first free licence ever created. It grants full exploitation rights and solely requires maintaining copyright notices and disclaimers of guarantees and responsibilities. It is a result of the distributions of versions of Unix by the University of California, Berkeley, in the seventies and eighties. The philosophy behind this licence is that the code is the fruit of the research and work of the University, financed by the Government of the United States (and the taxes of the American people). Therefore, it must be freely available and must protect what we refer to herein as the "moral rights" of the authors for the mere obligation to maintain copyright notices. The BSD has been the model for many similar licences, including the Apache licence and the licences of the X family (X, XFree86, XOpen, X11).

- **Free licences with strong copyleft.** The General Public License (GPL) is the most emblematic of the copyleft licences. Its purpose is to guarantee the four main freedoms of free software for all users and that any modifications be distributed under the same conditions. Others include the IBM Common Public License or the Sleepycat licence. They are known as strong copyleft licences as they do not allow their integration in major applications with other types of licences.

Free licences with weak copyleft

MPL, CDDL, LGPL.

- **Free licences with weak copyleft.** These licences maintain the copyleft obligations for the core of the program distributed under the licence, but also allow their integration in works with other licences. The Mozilla Public License (MPL), the Lesser GPL (LGPL), the Open Source License (OSL) or the Common Development and Distribution License (CDDL) are examples.

Free software rights

Looking at the rights that are granted under a free software licence:

- **Freedom of use**: The user of free software has full freedom to use and copy the software how, when, as much and where deemed convenient: install it on their hardware, store the necessary files and run it whenever they wish, in order to benefit from its applications. The user may use the software:
  - For any purpose or end. Therefore, the use of the software cannot be limited to the "personal use" of the user. Additionally, free software may be used for both private and professional purposes.
  - By anyone. Without there being room for discrimination due to the group of which they form part.

– On any hardware devices deemed convenient, regardless of their technical characteristics.

- **Freedom to copy**: users of free software may make as many copies of the software as they wish, without being limited to solely copying the files necessary to run the software on their hardware, or to a single security copy. This freedom to copy is closely related to the freedoms of use (users may use the software on any hardware devices they wish) and of distribution (the user may provide copies of the software, with or without modifications, to third parties).

- **Freedom to transform, and access source code**: users also acquire the right to transform the licensed software: translate it, adapt it to their needs, debug it or combine it with other programs. To allow users to effectively use this freedom of modification, the supplier- licensor must furnish them with the source code for the software or, at least, make it available to them. As defined by the GNU-GPL and the OSD guidelines, source code is the "preferred form" for a developer to make modifications to the software.
  Conditions on the freedom of modification of free software. Generally speaking, most licences impose some conditions on transforming: they must respect the copyright notice of the original author and sometimes must indicate which files they have modified. The purpose of this condition is to protect the reputation of the original author facing the possible malfunctioning of the software based on a modification.

Supplementary content

GNU-GPL Version 3 states in its Clause 5.a) that the "the work must carry prominent notices stating that you modified it, and giving a relevant date".

- **Freedom of distribution and public communication**. Free software users have the right to distribute copies of the software with or without modifications to third parties, in tangible form or over the net. This is a very broad freedom, inasmuch as the user may distribute them free of charge or in exchange for economic compensation, temporarily (rental, loan...) or indefinitely; with or without source code (copyleft requires access to source code), verbatim or modified.

GNU-GPL Version 3, no longer uses the term "distribute", but the more generic term "convey" (convey literal copies of the source code, convey works based on the program, convey the program in object code with the commitment to make available the source code, etc.), to encompass what we understand in Europe to be distribution and public communication. Other free licences, such as the Apache 2.0, not only include the licence to distribute programs, original or derived, but also to publicly display them).

It is as to conditions on distribution that free software licences most vary. As we have mentioned, permissive BSD-type licences only require users to maintain the "copyright notice" and disclaimer when redistributing the software, both in source code and binary. Copyleft licences require redistribution of the work and derivative works to be done on the same terms as the original code. Strong copyleft extends this to works that are combined with or intimately interact with the original work. In some cases, the free software licences contain certain limitations to the redistribution of the software, when such redistribution may conflict with a patent: for

instance, the duty of indicating that the free software is being sued for the violation of third-party patents, identifying such third party (Mozilla licence).

3.5. Freedom applied to works that are not software

Free licensing was initially conceived to be applied to software, but it also may be applied to other types of works. Upon careful study of the GNU GPL, it may be seen that the licence may be applied to information other than software. The GNU GPL holds that "it applies to any program or work containing a notice placed by the owner of the rights, claiming that it can be distributed under the terms of the General Public License". In this sense, the "program" must not necessarily be a computer program. Work of any kind subject to copyright may also freely be subject to copyleft under the GNU GPL.

The GNU GPL refers to the "source code" of the work; this "source code" implies different things for different types of information, but the definition of "source code" –as established by the GNU GPL– remains generic in any case: "the source code for the work represents the preferred form of making modifications to the work". However it sits awkwardly in relation to works other than software.

The FSF has further created a free licence for documentation, especially as the software is accompanied by technical documentation that is often necessary for its use. It would make no sense to distribute the free software without distributing the relevant documentation under similar terms. The General Free Document License was thus created to accompany their programs.

Creative Commons

The Creative Commons initiative (often abbreviated "CC") is a project created by experts in copyright law from Stanford University in the United States. Its purpose is to help authors and creators distribute their works for public use and thus extend the number of creative works available to all. It is especially directed to literary and artistic creations and not software, and expressly recommends the GFDL for any computer documentation and applications. The CC further provides a framework for dedicating works to the public domain, also under the conditions of the United States copyright laws.

Creative Commons is commented on in more detail below.

4. Free software licences

As mentioned earlier, the array of free licences spans from permissive licences, which impose no further obligations than that of attaching the conditions and the disclaimer, to licences with a strong copyleft, requiring that the same licence be maintained for redistributions of the software and of any derived work.

Along these lines, for the purposes of our study, we have classified free licences into three categories (plus one), to be examined in this section. These three categories are as follows:

- Permissive licences, BSD style, including MIT and X licences (compatible with GPLv2), and the AFL or ZPL (incompatible with GPLv2).
- Licences with strong copyleft: GPLv2 and GPLv3, in particular.
- Weak copyleft licences: LGPLv1 and the LGPLv2, the MPL and the OSL.

4.1. Permissive licences: no copyleft

In this section, we shall present some of the most commonly used free licences among the free development community, especially the BSD licence, which has served as the model for many other licences.

Supplementary content

Most are born from the academic world (as indicated by their names: Berkeley Software Distribution, MIT, Education Community License...), and therefore they have been referred to as "academic licences".

These licences are "at one end" of the array of free licences, as they do not contain copyleft obligations and allow for the privatisation of derived or collective works that include the software.

The first generation of these licences (BSD, MIT/X, Apache 1.0 and Apache 1.1) is characterised by being very short and not including any further obligations than those of maintaining the notices of authorship in the source files and the list of conditions (especially the disclaimer) when redistributing the software. The main objective of such licences is to grant the recipients full exploitation rights to the software (rights of reproduction, modification, distribution and public communication) so that the licensees may do "whatever they want" with the code. They do not contain copyleft and allow incorporating and combining the software with any sort of work, whether free or non-free. For instance, it is said that there are BSD software components in the Windows NT and Mac OS X operating systems.

The next generations (Apache 2.0 and AFL) include a series of new conditions relating to patents, governing law, etc., that are in line with the Mozilla Public License (which we shall discuss below), to modernise and clarify their terms.

4.1.1. Berkeley Software Distribution (BSD) and similar licences

The Berkeley Software Distribution (BSD) licence is perhaps the most simple of all free licences. It derives from the distribution of versions of Unix by the University of California, Berkeley, in the seventies and eighties, in the early beginning of the free software movement. The principle underlying the licence is that the software is the result of the college research and work financed by the Government of the United States (and the taxes of the American people) and that, therefore, it must be freely available. This means that it would only protect what we have referred to herein as the "moral rights" of the authors for the simple obligation of maintaining notices of authorship (copyright notice).

- Rights granted. The BSD allows unrestricted use, modification, copy and redistribution of software under the BSD, in object code (binary) or source code format.
- Obligations imposed. Distribution in form of source code is to be accompanied by a copyright notice, the list of conditions and the denial of any warranty and liability. Redistributions in binary code must reproduce the same things in the documentation. The name of the author and of the contributors may not be used for the promotion of derived works without their permission.
- Other terms. No warranty is granted in respect of the proper operation of the program and all liability is denied.

Therefore, almost anything can be done with codes under the BSD, provided the notice of authorship of the initial program is respected and the list of conditions is included in the code or documentation. It is also unnecessary to provide end users with the source code.

> The first version of the licence imposed the obligation to attribute each component to their original authors in any publication or promotional material of the program or derived work. This obligation implied certain hassles, as it was necessary to include extensive authorship throughout all the documentation and the source code, relating to each author adding their name to a licence. In a program with hundreds of contributors, this obligation was hard to meet. This also meant that BSD code was incompatible with GPL code. In July 1999, this obligation was stricken from the BSD licence. Regardless, at present, it is necessary to verify the version of the licence applied to BSD code, to make sure that it is not an earlier version and make sure that its terms are correctly followed.

The BSD-style licences allow for a great dissemination of the software and its use as a reference or standard (for protocols, services, libraries and even complete operating systems, such as Unix BSD). It nonetheless also allows what is known as code forking), inasmuch as anyone may adapt, modify and extend the program kernel and create a "similar but sufficiently different" version. This is seen, for instance, in the proliferation of operating systems with BSD-type licences, such as the OpenBSD, the FreeBSD and the NetBSD.

> Any software with a three-clause BSD licence (or new BSD) is compatible with GPL software (and almost any other free software licence), but not the other way around. In other words, BSD code may solely be incorporated in a GPL program (with the result of a work combined under the GPL), but GPL code cannot be incorporated in BSD software.

Other licences similar to the BSD

The BSD has been the model for many similar licences, among which we shall mention the MIT licences and those of the X family (X, XFree86, XOpen, X11), the Apache 1.1 licence (which we shall discuss below), Cryptix, Python, W3C Software Notice, Zope Public License (ZPL), LDAP Public License, Phorum, etc., and the OpenSSL and Sleepycat licences, which follow a simplified model of the BSD licence, but include copyleft clauses (as we shall discuss in the section on licences with copyleft).

The X and MIT licences are similar to the BSD licence but, on the one hand, they specify the permitted uses in further detail: "the use, copy, modification, merger, publication, distribution and/or sale of software"; and on the other, do not distinguish between distributions of source code and object code.

4.1.2. The Apache Software Licences (ASL)

The Apache web server project was created at the laboratories of the National Center for Supercomputing Applications at the University of Illinois, United States, and is now "run" by the Apache Foundation, in its technical and organisational, as well as its legal aspects.

The Foundation has drafted the Apache Software License (ASL), with versions ASL 1.0, ASL 1.1, and now, ASL 2.0, inasmuch as from January 2004 on, all software of the Apache Foundation will be published under ASL 2.0.

Supplementary content

Due to the importance that the Foundation and Apache software in general have in the free software community, in terms of software quality and its management model, the ASL is a licence that has been used by many other projects.

The ASL 1.1 is a variant of the BSD licence adding a few extra obligations:

- There is an obligation to maintain a notice with respect to the original authors in the documentation or redistributions of the software: "This product includes software developed by the Apache Software Foundation" (http://www.apache.org/)".
- Derived works should not use the Apache name without authorisation from the Apache Foundation (to maintain the reputation of the original authors).

Due to these additional obligations, the ASL 1.1 is not compatible with GPLv2. We should note that the first version of the licence (ASL 1.0) contained the same advertising obligation as the BSD with respect to the advertising materials mentioning the product.

The Apache 2.0 licence was published in January 2004 and belongs to a new generation of free licences. It is a very complete licence from a legal perspective, incorporating many of the modernisations contributed by the Mozilla Public License in 1998 (which we shall discuss in the following): complete

definitions, a patent licence and a patent peace agreement, an obligation to indicate modifications, a notice.txt file etc. It maintains its degree of permissiveness: it is not a copyleft licence.

- **Rights granted**. ASL 2.0 allows for the reproduction, modification, distribution and public communication (performance and display, under American law), with a right to sub-license, of the software under ASL in object code (binary) or source code format. Includes the explicit right to use another licence for the modifications or any derived work "as a whole", provided it meets the conditions of the ASL 2.0 licence.
- **Obligations imposed**. The redistribution of software should be accompanied by the licence, a notice if any files have been changed, any original notice of copyright, patent or trademark, any notice.txt file (with notices of authorship, modifications and any other legal notice). The name or trademarks of the licensor and contributors cannot be used.
- **Other terms**. No warranty is granted as to the proper operation of the program and all liability is repudiated. A patent licence is also included (revocable in case legal actions are brought based on patents against any other person with respect to the software).

Below, in the section dedicated to the Mozilla Public License, we will discuss the terms and objectives of the patent licence and the notice.txt file, as these concepts were created with this licence.

> As the Apache Foundation is a model for the management of free communities and projects, its new licence is an instrument used by many projects, especially those working with Java technologies or those of the Apache Foundation (Tomcat, ANT, libraries such as Commons, Jakarta, etc.). It is incompatible with the GPLv2, according to the FSF (due to the explicit patent licence) and it is considered that the ASL 2.0 is now compatible with the GPLv3. The importance of this licence lies in the express objective of the FSF to create a GPLv3 licence compatible therewith.

4.1.3. Other permissive licences

There are a number of permissive software licences that can be seen at the opensource.org website, and commented on the fsf.org website as to compatibility with the GPL. These include, among many others:

- Zope Public License.
- Open LDAP License.
- Artistic License 2.0: a licence modelled on the GPL but without a copyleft.
- Perl: a mixture of the GPL and the former Artistic licence.
- Academic Free License 3.0: A "complete" permissive licence on the MPL model.
- Python 2.0.1, 2.1.1 and later versions: A BSD-style licence, requiring that the program should be subject to the laws of the state of Virginia.
- PHP 3.0: A BSD-style licence including the obligation to incorporate a PHP advertising clause.

- Q Public License (QPL)1.0: A licence compelling the distribution of any modification as a patch to the initial program. It is also necessary to refer to the initial supplier (Trolltech) any modification not available to the public.

4.2. Licences with strong copyleft

Above, we have explained the concept of copyleft from a legal perspective: the obligation to use the same licence for the redistributions of software, with or without modifications, or of a program containing the original software. In this section, we will explain in detail two licences with strong copyleft: the GPL v2 and v3.

We shall see that almost all licences with copyleft are incompatible among each other, as they all require the use of the same licence for redistribution, which gives rise to a conflict in respect of which is to be applied for a program mixing two components under different copyleft licences.

4.2.1. The GNU General Public License, version 2.0 (GPLv2)

Created in 1989, the GPLv2 has been described as being "part licence, part political manifesto": its preamble contains a description of the free software principles and a simple summary of the licence; the main part specifies the rights granted to the users and the limitations and conditions imposed on the exploitation of the software.

> It is important to stress that in spite of its familial and simple tone, the GPLv2 was designed by Richard Stallman with his American legal counsel and therefore does not contain any old provisions, but a deliberate and very subtle means for licensing and conditioning the exercise of copyright rights.

Following we have presented the GPLv2, due to its importance, in some detail.

Useful definitions. Although not explicit, as in the MPL or the CPL, and now in the GPLv3, the GPLv2 contains several definitions that are of great use (and sometimes confusing):

- *Program*: any program or work to which the licence has been attached. Technically, this could include a text, image or other file (clause 0).
- *Work based on the program*: the original program or any derived work thereof, according to the copyright definition. This would include any work containing the program or a part thereof, whether a true or literal copy, or with modifications (clauses 0 and 2).
- *Source code*: the preferred form of the work to subject it to modifications. Regarding an executable file, the obligation to provide the source code includes all modules contained thereby, plus the configuration of the interface and scripts to control compilation and installation. Does not include the source code of the equivalent modules of the operating system

in the program being run, unless such modules accompany the executable (clause 3).

Rights granted by the licence. These guarantee the four main freedoms:

- The right of reproduction and distribution of the original source code (clause 1).
- The right of modification of the program or a part thereof (clause 2).
- The right of distribution of the source code and the future modifications, provided they are distributed with the same GPL and without charging (clause 2b –the copyleft clause–).
- The right of reproduction and redistribution of the program (and its modifications) in object code or executable format, with the same copyleft condition and provided it is accompanied by the source code or the source code is made available to a third party, without charging anything other than the cost of delivery of such source code (clause 3).

Access to the source code is the second fundamental aspect of the licence. A program may be distributed with the GPL in binary (object code) format, but it must always be accompanied with the source code or the offer to provide it to any third parties for a term of three years (clause 3).

Obligations: the GPL contains several conditions and limitations:

- Any distribution of the program or of work derived therefrom must be accompanied by the notices of authorship, an indication of any modification made (and its date), the disclaimer of warranties and a copy of the licence (clauses 1 and 2).
- It is not allowed to copy, modify or distribute the program in a manner other than that expressly permitted by the licence, with less freedoms or greater restrictions (clause 2b and clause 6).
- If any act in violation of the licence is attempted, the licensee shall waive their original rights (clause 4).

Versions. The licence allows authors-licensors to refer to new versions thereof (we shall discuss version 3.0 below) adding that the work is published "under version 2 and any subsequent version" (clause 9). This flexibility allows that programs maintain compatibility with future programs under a subsequent version of the GPL –such as the recently published GPLv3. In this case, licensees may choose the applicable version.

Supplementary content

Linus Torvalds, for instance, has excluded subsequent versions for the Linux kernel: he is staying with version 2.0.

Other points:

- Warranties. Clauses 11 and 12 clarify that no warranty is offered in terms of the proper operation of the software covered by the licence and repudiate any liability for damages. We have nonetheless seen that the validity of these clauses is dubious (in jurisdictions other than that of the United States and even in the United States in some circumstances) in the light

of the consumer protection laws and the prohibition of abusive clauses in accession agreements.

- Governing law. The GPLv2 does not include any clause indicating the governing law or the courts of competent jurisdiction to try any conflicts in reference thereto. Therefore, the relevant law would be applied at the corresponding courts under the principles of conflicts of law. In most cases, it will be the law of the legal domicile of the licensor, but a consumer, for instance, may choose the law and courts of their domicile.
- Patents. The last paragraph of the preamble stresses the dangers that patents pose for free software. The GPLv2, nevertheless, does not include any clause restricting the possible patents on software under the GPLv2 or requiring their licensing in favour of other users (the GPLv3 does). As a logical consequence of the obligation to distribute the program and any work derived therefrom in terms equal to those of the GPLv2 (clause 2b), any licensee obtaining a patent on software under the GPL must allow its free use under the GPLv2 by all subsequent recipients – which could be considered an implicit patent licence. We shall later see that the GPLv3 specifies the terms of the patent licence.

Comments on the GPLv2

An important matter which must be clarified is the matter of **derivative works** and the application of the GPL to them. It is a key concept in understanding the GPLv2, as it defines the scope of the copyleft clause, which is what most distinguishes this licence from other free licences. This matter has given rise to a great deal of controversy in the world of free software and software in general.

> We have already said on several occasions that software cannot be "privatised" under the GPLv2, nor may its derived works. Therefore, some developers doubt to incorporate or relate their work too closely to a copyleft program, as they fear losing them under the GPLv2 in circumstances in which they cannot or do not wish to permit it (such as a non-free development or different free licence).

What does a derived work or work based on a program consist of, according to the authors of the licence? The definition cited earlier refers to the definition under copyright law: it is work containing the program or a portion thereof. But the word contain in the field of programming, leaves room for doubt: are we dealing solely with derived works under a strict legal interpretation of copyright or author's rights? Or does it also apply to "composite works" or collective works, incorporating the original program?

Software components may interrelate in many ways, by various sorts of calls or links. The compiling of a program (to create an executable) may incorporate several components in a single program, or the various components may interrelate when the program is interpreted when run. Each such interaction could have different legal effects. What is debated is whether these architectures imply that the resulting work would be subject in whole or in part to the GPL. This issue has become more complicated with the evolution of program-

ming methods (structured or by objects) and computer languages (C, C++, Visual Basic, Java, PHP, etc.), many of which did not exist upon drafting the licence.

Suggested reading

Slashdot: D. Ravicher on open source legal issues. http://slashdot.org/interviews/01/06/05/122240.shtml

M. Assay. A funny thing happened... www.linuxfordevices.com/files/misc/asay-paper.pdf

L. Rosen. The unreasonable fear of infection. www.rosenlaw.com/html/GPL.PDF

Clause 2b itself states that copyleft applies to any work *containing or derived from the original program*, which must be licensed as "a whole" (with all of its components) under the GPLv2.

- We should first note that the sole gathering or putting together of a work (separable, not based on un a GPL-covered program) on the same medium with GPLv2 software, for distribution for instance, does not imply that such other work must be distributed under the GPLv2. The licence further clarifies that if the identifiable parts of a work could be considered to be individual independent works in themselves, the licence shall not apply to such parts.

- Facing other cases, prudence tells us that it is necessary to assess the risks relating to a particular development or architecture, considering the design and potential consequences of being subject to the GPL. We can say the following with some certainty:
  - If, when a new development is compiled with a GPLv2 work, the final executable includes elements of the original program (in the case of components with static links between themselves), then the modifications may be considered separable and, consequently, the entire work and each of its parts must be distributed under the GPL.
  - If the original GPLv2 program and the new development coexist separately (even when contained on the same medium) and the particular development calls the GPLv2 program in run time (the case of a dynamic link), unfortunately, the situation is not so clear. The interpretation of the FSF is certainly that dynamically linked works, and other forms of interaction such as plug-ins, would lead to relicensing under the GPL if the degree of interaction is sufficiently "intimate" or dependent.

Among the "frequently asked questions" of the GPL, there a few explaining cases of modifications, links and calls to GPL code that the FSF "resolves" by offering its interpretation of the licence and the law. For instance, it is clarified that a new program compiled by a compiler under GPL shall not need to be distributed under such a licence, except if the executable resulting after the compilation incorporates elements of the free compiler or other GPL program.

But the subject is not completely resolved for the GPL and, in the end it is left to the judgment of the creators of modifications and derived works to consider whether they are subject to the GPL (and when to consult with legal counsel).

> Linus Torvalds has expressly included in the GPL covering the Linux kernel of the SO GNU/Linux, an addendum to state that he, as licensing author, does not consider that programs with dynamic links to the kernel are subject to copyleft. User applications and other non-core elements of an operating system, such as drivers, interact dynamically with the components and the modules of the system kernel. Therefore, the applications and controllers are specific to one platform or the other. There is a possibility that such interaction with a GPL operating system could affect such programs and drivers. Without this clarification, almost any program run on GNU/Linux and with calls to its central libraries could be considered, based on the strictest interpretation of the licence, to be subject to the GPL. This would reduce the use and dissemination of GNU/Linux as an operating system to an environment of programs compatible with the GPL.
>
> Nonetheless, over time, L. Torvalds seems to have evolved towards an interpretation closer to that of R. Stallman...

Translations. There are no official translations of the GPLv2. In other words, the original English version shall be that determined by the terms of distribution when the original GPL is applied to a work. There are unofficial translations indicated on the pages of the FSF, which it does not approve as legally valid. It should be noted that if an author applies a translated GPL to its program, the translation of the licence should prevail, not the original GPL in English (except as otherwise indicated). Should there be a translation error, the results could not only be unpleasant, but also horrific for the free software community. There would be "quasi-GPL" versions and modifications of software (with foreign hues) mixed together with true GPL programs (in their English version).

Compatibility of other licences with the GPLv2. A program is compatible from a legal perspective with software under the GPLv2, when distributed in terms that are compatible with those of this licence. They cannot be more restrictive (as in the case of any non-free licence), but may be more permissive (as in the case of the modified BSD licence or the LGPL, which we shall study hereafter).

> Compatibility with the GPL has the dual advantage of facilitating the integration of free components in more complex and integrated distributions and platforms, and ensuring that the code may be integrated fearlessly with 75% of the free software programs available over the internet.

Note that GPLv3 is not compatible with the GPLv2 (but it is with software under the GPL2 "and later versions"), making it necessary for the owners of software under the GPLv2 "alone" to relicense it upon the terms of the GPLv3 (or a more permissive licence) if they wish to ensure such compatibility.

Some examples of incompatibility with the GPLv2

- The Clause of the original BSD licence and the Apache 1.0 licence requiring the inclusion of a mention of the original authors in any advertising or promotional material of the program.
- Clauses reserving rights of the Netscape Public License, allowing Netscape to benefit from third-party modifications to the Navigator and incorporate them into new Netscape products.
- The explicit ASL 2.0 patent licence (in the opinion of the FSF).
- The obligation to obtain a "developer's licence" to be able to integrate Qt elements in applications that are not Windows X System, provided by the Qt licence.

4.2.2. Version 3 of the GPL

The process of modernisation of the GPLv2 began in 2005 and ended on 29 June 2007, when the FSF published the new GPLv3. Such modernisation responds to various needs, among the main of which are the following:

- Licence internationalisation.
- Improved flexibility.
- Response to author's rights management systems (DRM) and their legal protection.
- Management of legal issues relating to software patents.

To these four items, we could add one more: clarifying the scope of copyleft with respect to new technologies and architectures, dynamic links and the concept of source code.

The main differences with respect to the GPLv2 are discussed below.

a) Definitions. First of all, besides a new definition of Program, You (user) and Modify, there is a new definition: "*complete corresponding source code*" (Clause 1) and two new terms: *propagate* and *convey* (Clause 0).

- The scope of the definition of source code is important, due to the obligation to distribute or offer access to the source code (under the GPL) of any executable distributed without them (GPLv2, Clause 3).
  - GPLv2 defines source code as "the preferred form of the work (program) for making modifications to it" and the obligation to provide the source code includes any "script necessary for compiling the program".
  - In GPLv3, the definition of source code is the same, but the relevant obligation refers to the "complete corresponding source code", which is, a priori, much broader: it includes the "code necessary to generate, install, run and modify (the program)"; the scripts for performing these operations and definitions of interface and (explicitly) the source code of shared or dynamically-linked libraries that the program is designed to use.

- The terms propagate and convey are used, according to the purpose of internationalisation of the licence, to cover all acts reserved by copyright

under any legal system, without mentioning such words as distribute or reproduce, which could be legally defined differently in various jurisdictions.

– Propagate is used to designate "any activity requiring authorisation from the owner of the program", except the running of the program and private modifications (i.e., those not destined for third parties).
– Convey is a subgroup of propagate for the purposes of copyleft obligations (which would activate with the "conveyance"): it means to perform an act of propagation resulting in the creation or obtaining of copies by third parties; for instance, the delivery of a copy to a third party, public communication of the software over the internet, sharing it on P2P networks, etc.

b) Rights granted. While the GPLv2 indicates no authorisation is required from the owner to run the program (considering that the "use" of a program is not subject to copyright), the GPL3 expressly grants:

- The unrestricted right to run and modify the program for private purposes.
- The unrestricted right to propagate the program, provided it does not result in the conveyance of the software. This would therefore include the right of reproduction, modification and internal "distributions". It also allows the delivery of the software to third parties unconditionally, when done under a consulting agreement whereby the consultant is to make modifications exclusively for the licensee (work-for-hire).
- The right to transfer the software under copyleft conditions.

c) Obligations. The basic obligations with respect to the copy and the distribution of the software are similar to those established in the GPLv2: it is necessary to maintain notices of authorship, the licence, notifications of changes, etc. If the program has a user interface, it must contain a system for publishing copyright notices, the disclaimer and the access to the licence –an obligation stronger than that of the GPLv2.

Regarding the copyleft system, the GPLv3 does not change much either:

- It maintains the obligations to convey any modified work "as a whole" "under the same licence" (letter 2b of the GPLv2, now 5c).
- It slightly modifies the obligation to accompany any distribution of binary code with the "complete corresponding source code" or offer access thereto to any third party who has the binary. The term of this offer is the greater of three years, or the duration of any medium or offer of "corrections". The cost of its distribution may also be charged.
- It specifies five ways to make this distribution/offer (such as, for instance, distribution on CD, from internet servers or sharing on P2P networks).

d) DRM. In Module 2 we discussed the legal system of protection of copyrights management systems (Digital Rights Management or DRM): it is illegal to "circumvent" (i.e., crack) an effective technological measure, capable of protecting author's rights. The GPLv3 has two mechanisms against those systems, which it considers a violation of the freedom of the users (the FSF calls them Digital Restrictions Management):

- On the one hand, its Clause 3 states that by no means shall GPLv3 software be considered part of an "effective technological mechanism of protection" of rights and that the owners waive the right to sue third parties for any act of elusion resulting from the mere exercise of the rights assigned under the licence. By these indirect means, it seeks to allow that any GPLv3 software be modified without infringing upon such rules, which would prohibit that type of "circumvention". The consequence sought is that it will be incompatible to distribute GPL3 software on DRM programs whose licence does not allow access, modification or reengineering. Whether this works legally is a subject of debate, especially considering the imperative nature of the system of protection of these DRM systems.

- On the other hand, the GPLv3 includes, in the definition of "complete corresponding source code", exceptionally for consumer products, the access and deciphering keys and the information for installing and running modified software. With the GPL3, manufacturers and distributors of "closed" devices for users / consumers cannot prevent access to the device or demand obtaining payment for a key, for instance, to "access" or run the device or modify its program code. If they did, they would also have to surrender the keys, codes and the relevant information.

e) Patents. The patent protection system in the GPLv3 is complex, due to the various practices that have arisen in terms of software patents. Under GPLv2, any assignment of patent rights (to a process implemented with GPL software) was implicit, with the consequent uncertainties in terms of its legal effects. In GPLv3, there are four important terms (Clause 11):

- The assignment of patent rights is made explicitly: if someone has a patent on their contribution to software distributed under GPLv3, it grants a patent licence to use, market and import the contributed software to anyone using such contribution without modifications.
- Any explicit patent licence granted to a licensee shall be extended to all licensees.
- Additionally, a "cascading" protection mechanism is sought to be established: those distributing software under GPL3, benefiting from a patent licence from a third party, must extend its benefit to all licensees, or waive the benefit, or guarantee that the "corresponding source code" is available to all under the conditions of GPLv3.
- Regarding the agreement between Microsoft and Novell of March 2007 (not to be covered by the licence), if someone obtains specific protection

in respect of software under the GPLv3 that, in a discriminatory manner, may solely protect them and their licensees, such software cannot be transferred under GPLv3.

f) Remote services or Application Service Providers (ASP). It was thought that the new licence would restrict the use of GPL software by those offering commercial services to their end users based on GPL software, without distributing their programs and sources (Google and Yahoo! are obvious examples) or that they would be compelled to furnish the source code of any ASP service. In the end, this mechanism has been left for the Affero GPL and an explicit compatibility is included with the licence.

g) Additional permissions. The GPLv3 allows adding some additional permissions (but not restrictions), such as exceptions from its obligations. These shall apply to identified software components and may be eliminated by the licensees upon redistribution. The LGPLv3 is an example of this, as it consists of the GPLv3 with the additional permission to link to programs "using the library" under any licence (as we shall see hereafter).

h) Additional restrictions: licence compatibility. The "legal compatibility" of the software is fundamental in the development of free software: it means being able to mix two programs with different free licences, without either being in breach in redistribution. The GPLv2 prohibits adding any additional restriction not included in the licence itself. This has led to licences with agreements in respect of patents, attribution of authorship, use of trademarks, notices and disclaimers with differing terms, being declared "incompatible" with the GPLv2 by the FSF (and by attorneys advising their clients). The GPLv3 makes an effort to enhance the set of free licences compatible therewith through a new mechanism: allowing the addition of six types of additional restrictions on programs or code added to the GPL3 code.

The restrictions are compatible if they refer to:

- Maintaining notices of authorship or other forms of attribution (for instance, notices of "powered by" or "about" windows) and obligations to indicate any modification made to them.
- Disclaimers (warranty exclusions and liability limitations) in terms other than those of the GPL3.
- How to indicate modifications.
- Restrictions on the use of the names of authors for advertising purposes (the former BSD licence continues to be incompatible).
- Granting rights or prohibitions in respect of the trademarks.
- Indemnities for contributors.

The Apache 2.0 licence is an example of licence that is now GPLv3 compatible.

4.2.3. Other licences with strong(er) copyleft

While the GPL is considered to have (debatably) the strongest degree of copyleft, encompassing both derivative works and works which, on a wider interpretation, could be considered based or dependent on the GPL code (or contain it), other free software licenses have a strong copyleft effect.

Common and Eclipse Public Licenses. The CPL and the EPL (and their predecessor, the IBM Public License) are legal instruments developed by IBM, with a format differing from that of the GPL and the BSD, the two predominant models. The CPL is closer to the Mozilla Public License, as it has a more "legal-like" form (including definitions and governing law) and covers such issues as indemnities among contributors and patent licences. They are well drafted licences from a legal perspective and leave much less room for doubt than the GPLv2, for instance. Definitions are clear, as is the scope of the rights and obligations. Our main comment is that the licence is incompatible with the GPLv2 due to the obligation to license any patent of the contributors and compensate the co-authors in the event of claims by commercial users (cross-indemnity among contributors). A priori, we understand that this continues to be incompatible with the GPLv3, although it too has a quite similar patent licence, as regards commercial indemnification.

Aladdin Free Public License (AFPL) The Aladdin Free Public License (AFPL), relating to Ghostscript, warrants special mention as it has a particular nature. It does not comply with OSD, although it is directly inspired by the GPL. What is interesting is that, while the latest available version of Ghostscript is distributed under the AFPL and requires obtaining a non-free licence for commercial uses, the penultimate version of the software is released under the GPL. Therefore, the "best" version of the program is marketed and free developers may take advantage of the oldest code.

Sleepycat Software Product License (Berkeley Database). This is a licence applied, most of all, to a database engine of the Sleepycat corporation (formerly Berkeley Database). It follows the simple model of the BSD licence, which we shall discuss hereafter, and adds an obligation to distribute or make available the source code of the software and of any other program using the software. Such a program must also be freely redistributable under reasonable terms (copyleft). Open and free licences are considered reasonable, as is the GPL.

GPL Affero 1.0. Affero is software for managing and extending virtual communities with rating and e-commerce functions. The licence is a variation of the GPLv2, drafted with the aid of the FSF. The licence covers the case of the architecture of programs distributed on networks or services linked by web services. In this case, the user / licensee does not receive the program as software distribution, but as a web service, and may offer the same service to third parties, avoiding the copyleft obligations of Clause 2b. The Affero licence adds to the GPLv2 a Clause "2d", which provides that if a service offered over the

web by the original program were to have a function to provide the source code also over the web, the licensee cannot eliminate that function and must offer access to the source code of the derived work over the web.

Affero GPLv3. The new Affero GPLv3 licence is basically the GPL with an additional agreement to cover the same scenario as mentioned with respect to Affero 1.0. In this case (ASP), users of remote services must be granted access to the source code. The GPLv3 is expressly compatible with Affero GPLv3 and vice versa.

Licence OpenSSL / SSLeay. This licence applies to SSL security programs. It is a combination of the Open SSL and SSLeay licences. It is modelled on the BSD licence and adds to the end of the SSLeay licence a copyleft clause requiring that any derived work be distributed upon the same terms. Mixing this code with GPL code is expressly prohibited. It is also incompatible with the GPL inasmuch as it has a clause with respect to advertising and the attribution of authorship (derived from the earlier version of the BSD and the Apache).

4.3. Licences with weak copyleft

In this section we shall discuss free licences that are known a having a weak copyleft effect: they are distinguished from strong copyleft in that they allow for their integration, use and redistribution in programs subject to other licences, but maintain their own code subject to copyleft.

4.3.1. The GNU Lesser (or Library) General Public License (LGPL)

The GNU Lesser General Public License (or Library GPL) is the second licence drafted by the Free Software Foundation. Initially, this licence was known as the "Library GPL", as it was designed expressly to be applied to computer libraries.

The FSF later changed its name to "Lesser GPL" as it considered that it guaranteed less freedom than its older sister, the GPL. Its version 2.1 is of February 1999 and, in June 2007, version 3.0 was published, which is a variation of the GPLv3, discussed above.

In the preceding sections we have mentioned that when a program links to a software component, whether it be statically or through a dynamically-shared component or API, the combination is considered a work "based on" or "derived from" the original software. If the software is under the GPL, many argue that this link would force distribution of the entire final program under the GPL. The LGPLv2 was created specifically to allow certain free software components –libraries – with non-free programs, without affecting the resulting program. Therefore, a library with LGPLv2 offers a certain comfort or certainty for the developers of non-free applications wishing to link their programs with components under free licences, but that fear the copyleft effect of the GPL.

Supplementary content

The LGPLv2 derives from the GPLv2 and most of its terms are similar to those thereof. We therefore refer to the section on the GPL (both version 2 and version 3). Here we shall solely comment on its distinguishing elements.

As for the GPL, the LGPLv2 defines program and source code. It also includes three new definitions:

- **Library**: consists of a series of software components destined for linking with programs (using the functions incorporated in libraries) to create an executable.
- **Library-based work**: contains the definition of program in the GPL and means the original library or any derived work thereof, according to the definition provided by copyright law, i.e., work containing it or part of it.
- **Work using a library**: is separate work containing no part or derived work of the library, but rather is destined for being run with the library through compilation or links.

Regarding the same library and its modifications, the conditions applicable are those of the GPL. The main difference with the GPL is that the LGPL allows for the unrestricted distribution of an executable, consisting of the compilation, on the one hand, of works using the library and, on the other, the library itself (Clause 6). This is the exception to the regular copyleft clause of the GPL.

> Nonetheless, the recipient must be allowed to modify the program (even the work "using the library") for particular use and for performing reverse engineering operations to correct errors (therefore, it is argued that although there is no copyleft, it remains necessary to provide the source code).

As an additional condition, the LGPL applied to its library may be converted into the GPL at any time (there is no turning back) (Clause 3). We should also note that the LGPLv2 is compatible with the GPLv2, but not with GPLv3 or LGPLv3.

Due to its language, the LGPL is destined for use by libraries. But its use is not restricted to them, as there are other programs distributed with this licence (for instance, OpenOffice.org). The authors of the software are free to use the licence of their choice, regardless of their program.

> The FSF no longer recommends the use of the LGPL, except for strategic reasons: the use of the LGPL allows the broader distribution and use of its code and, therefore, favours the establishment of a component –a library, a program module etc.– as the standard in the sector. The LGPL does not, however, favour the development of free applications, which is a fundamental objective for the FSF, and therefore does not receive its full approval.

As a practical comment, we should note that, within the limits of the technical matters of the type of link between two programs, it is possible to combine, integrate and distribute libraries under the LGPL with software under any other licence, even non-free. An example of this type of software is the C library (libgcc) distributed with Linux, which may be used to develop non-free programs running on Linux.

LGPLv3

LGPLv3 is an explicit variation on GPLv3, i.e., it is GPLv3 plus additional permissions. Such permissions authorise the use of the library in question by a third-party program and licensing "as a whole" under a licence other than the LGPL. It also does not apply Clause 3 on DRM systems.

4.3.2. Mozilla Public License

The Mozilla Public License (MPL) was developed along with the Netscape Public License in 1998, when Netscape "opened" (as open software) the code of its internet browser, Netscape Navigator. The development of the licence was a collaborative effort between several of the "gurus" of the open movement, such as Linus Torvalds, Bruce Perens and Eric Raymond. They initially sought to persuade Netscape to use the GPLv2, but facing the refusal by Netscape and the need to respect the intellectual property of third parties, they ended up distributing the code under the NPL.

> Consulting with the community. Before opening its source code to the public, Netscape distributed a draft of the proposed licence on a newsgroup created especially to gather opinions on the matter (netscape.public.mozilla.license). The process of open development for the software carried over to the free world and awoke great enthusiasm... and criticism. There were several proposals to modify some of the terms of the NPL, especially that which allowed Netscape to use the same code in other products not under the NPL. This process has been followed by the Free Software Foundation in drafting the GPLv3.
>
> In the end, seeking balance between the commercial and free development objectives of Netscape and the free community, it was resolved to issue two licences: the NPL and the MPL. The first was applied to the initial code of the Navigator and to the modifications made thereto, and is no longer used. The second was applied to any software added to the code and to any completely- new program wishing to use the licence. The MPL is now used for several programs, including the Firefox navigator and other programs from Mozilla.org. The two licences are identical, except for some rights reserved by Netscape in the NPL for its initial code, which is only of "historical" value.

The MPL has a classic software licence structure and begins with important definitions permitting, among other things, distinguishing between what is original code and what is added code.

- **Initial developer**: in the case of the NPL, Netscape; in code under the MPL, the initial author indicated in the annex to the licence and any author of contributions.
- **Initial code**: code distributed by initial developers.
- **Modification**: any modification to the covered code not including a simple addition of a new separate file or a new code acting with the original code without modifying it (for instance, through an API –even if the API itself could be a modification, if integrated in the covered code. The word "modification" does not refer to the entire new work (as is the case with the GPL), which may also be a "derived work" under the law, but rather refers solely to the modified part.
- **Covered code** (covered by the licence): initial code plus modifications.
- **Contributor**: any third party modifying the covered code.
- **Larger work**: a work separate from the covered code but that may incorporate it or may link to it, without modifying it (Clause 3.7).

The meaning of "modification", summarised here, clarifies many things that the GPLv2 did not make clear –especially the matter of additional new files that do not modify any part of the initial code at development. It therefore allows a developer to add separate files and programs (non-free or free) and distribute them separately from the covered code, but as part of a larger program (potentially non-free).

a) Rights granted. As with all free software licences, the initial developer, first, grants a licence for the free use, reproduction, modification and distribution of the code and, second, a patent licence that is sufficient to allow the use of the program and modifications (Clause 2.1).

- Each contributor provides similar licences in relation to their contribution or modification (Clause 2.2).
- The code may be distributed in binary under a licence compatible with the MPL, provided the obligations contained in the licence are respected, such as access to the source code, for instance (Clause 3.6).
- The covered code may be included in a "greater work" (including it, but not modifying it) under any licence, provided the obligations relating to the covered code are respected (Clause 3.7), for example, access to source code.

b) Obligations. The source code of the initial code and any modification (covered code) must be distributed under the MPL, without more restrictive clauses (copyleft for the covered code, Clause 3.1). If the covered code is distributed in binary, access to its source code must be offered to the recipient of the distribution for at least twelve months (Clause 3.2). It is necessary to accompany any modification with a copy of a licence and an indication of the modifications and their authors, and indication of any known claim to the code (legal.txt) (Clause 3.3-3.5).

The MPL is a complete licence –imitated to some extent by the CDDL, the CPL, the OSL and now, dare we say it, the GPLv3. It is a much more clear and complete licence than the GPLv2 and, evidently, than the BSD. It was drafted with and by attorneys in the context of a commercial company and thus includes specific definitions and contains traditional matters relating to licences, such as competent jurisdiction and governing law.

Although its effect could seem closer to the BSD than the GPL, there are several important matters that we must consider and we shall discuss in this section:

- The MPL has partial reciprocity or copyleft, as does the LGPL: the covered code (including any modification) must be kept under the MPL, while any extension (larger work) may be non-free. It is also very easy to create an additional non-free file calling the original code under the MPL and distributing it entirely under a non-free licence. This continues with the philosophy of the BSD licence. Nonetheless, in all cases, the source code of the original free part must be distributed or offered to the recipient. This may all be illustrated as follows:

- Any software under the MPL 1.0 (and the MPL 1.1 with no alternative licence) is incompatible with the GPLv2 and the GPLv3; fundamentally as it contains too many additional restrictions relating to patents (although the GPLv3 is close in that aspect) and the possibility of linking to non-free programs, among other things. The possibility of multiple licences offered by version 1.1 allows compatibility if the GPL is chosen as an alternative licence (the source code of the programs of Mozilla.org, for instance). Figure 3. Illustration of persistence of the MPL.

- Patent clauses. As we have already seen in relation to the GPLv3 and the CPL, the termination clause (in this case clause 8), combined with the patent licences (Clause 2.1), is part of a new generation of clauses in free licences to create a work environment free of patents and free of the risk of patents. It constitutes what is known as "patent cross licensing". Developers cannot prevent a person from requesting and obtaining a patent on a process that may be part of a modification of the initial program (in the United States). The risk is that the use or a subsequent modification of the software could infringe upon a patent if the user does not use an appropriate patent licence.
  These clauses therefore seek two things:
  – *On the one hand, the "patenting" person must grant all other licensees (users and developers) a patent licence with respect to the patented process or code included in their contribution.*
  – *Additionally, the licences of author's rights (and patents, if any) granted to such "patenting" person shall be cancelled in the event of any litigation or attempt to prevent the free exploitation of the modification.*

- Commercial balance. The concepts of modification and larger work have been carefully prepared to find a balance between the freedom of the BSD, allowing an unlimited use of the code and the freedom of the GPL, requiring that all code and free derived works should be maintained, i.e., between the promotion of the development of free software by commercial companies and the protection of the work of "free" developers. This fair mid-ground has been defined by the difference between a modification and an addition. We should bear in mind that the GPL, in contrast, affects the additions intimately linked to software under the GPL.

- *legal.txt*. This is a file where the contributors must include notices of any claims, litigation or restriction on any part of the code. It evidences a clear knowledge of the process of free development, in which the risk of claims relating to intellectual and industrial property is high and transparent information is essential. A subsequent developer must use this file to study the legal limitations of code provided by third parties, perhaps in relation to a patent litigation, perhaps due to the limitations of certain parts of the code that may be under a licence that is compatible but differs from the MPL...

4.3.3. Open Source License (OSL)

The Open Source License (OSL, now version 3.0) is a licence with a weak copyleft, drafted in a neutral manner by the legal advisor of the OSI, Lawrence Rosen. It is a complete licence (definitions, licence explicit to the various rights, etc.) and conforms better than others to the legal framework of intellectual property in Europe and limitations regarding warranties and liabilities.

The OSL 3.0 limits its copyleft effect to *derived works according to the intellectual property law* applied in each case. It is argued that the GPLv2 is sought to be extended beyond what is permitted by author's rights alone (reproduction, modification, public communication and distribution) and could be limited by a strict interpretation of the law. The scope of the copyleft of the OSL is strictly within the scope of exclusive rights of the authors under intellectual property.

> This would allow, for instance, the linking software under the OSL 3.0, as libraries or with dynamic links, and the licence would not "affect" the software using such libraries, to the extent that they were not "derived works" of the original software.

Beyond the copyleft provisions, the definition of governing law and competent jurisdiction (in favour of the licensor) is more favourable for the authors and software distributors. Additionally, with an express warranty of title to the software and coverage in respect of wilful misconduct and personal damages, the warranty and liability limitations shall be more valid in Europe. Finally, distribution within a group of companies is not considered distribution for the purposes of copyleft obligations, as is the distribution of the services provided by the software (in ASP or "SaaS" mode) in which case it would be necessary to provide the recipient of the services a copy of the source code.

4.3.4. Other licences with "weak" copyleft or "hybrid"

There are a number of other free software licences following the weak copyleft model or tenets of the LGPL and the MPL. Each licence has been created for a specific or generic purpose, and must be understood and chosen in accordance with its own wording and merits applied to the specific case.

Apple Public Source License v. 2: A variation of the MPL created by Apple, with new elements, such as governing law (California), and covering the possibility of offering services over the internet (externally deployable), similar to the Affero.

CDDL: This is a generic version of the MPL created by Sun Microsystems with some modifications and without the commercial name Mozilla. Used for OpenSolaris, among other programs. The main differences with the MPL is that it does not include "scripts for the creation of executables" or API, etc., in the definition of source code. In case of distribution of the binary, the source

code must be generally published (not limited to distribution recipients). The *legal.txt* file of the MPL has been eliminated. The patent peace is limited: the patent licence is revoked in case of claims based on patents with respect to processes implemented by the covered code. Governing law is flexible, defined by the original owners. Copyleft includes distributions of services of the program to clients in ASP mode (sources must be offered to the service recipient).

EUPL 1.1. The European Union Public Licence is a new licence (of January 2007), expressly drafted for the release of software by the European Public Administration and the member countries of the European Union. The scope of copyleft is similar to that of the OSL, it contains a patent licence and the limitations on warranties and liabilities are valid within the general consumer protection framework and the accession agreements of the European Union. To establish an express compatibility with other copyleft licences, it contains a compatibility agreement with other licences included in an attachment (currently the GPLv2, the LGPLv2, the OSL, the CPL and the CeCiLL, a French copyleft licence): in case of mixing software under the EUPL with software under such licences, the software could be distributed under the new licence. The European Commission has published official translations in the languages of the European Union.

eCos licence 2.0 and Classpath. This is an FSF licence on the Embedded Configurable Operating System. It basically consists of the GPL plus an exception that allows linking the program to other programs that are not under the GPL and with effects quite similar to the LGPL. Whether integrated by compiling or linking to a non-free program distributed in binary, the eCos source code must be provided or made available. Classpath contains the same exception. What is interesting to note is that Sun has published a large part of the Java platform under the GPL, with the exception of Classpath.

CPAL: Common Public Attribution License is a variation of the MPL, with an explicit "Attribution clause" that requires publishing either on the user interface or another manner, attribution to the original developer of the code.

4.4. Other "free" licences

In the previous section we studied in depth the main free licences and discussed their features, their compatibility and consequences. In this section, we wish to complete our analysis of "free" licences. We will comment, in order, the following:

- Licences that we shall refer to as "pseudo-free", seeking to emulate free licences but containing a restriction that does not meet the freedoms of the FSF or OSD guidelines.
- Free documentation licences.
- Freeware and shareware licences, which are by no means "free".

4.4.1. The rise and fall of "pseudo-free" software licences

Although in this module we have focused on free software licences, it is interesting to present a brief analysis of the licences created by commercial enterprises seeking to benefit from a free development model –without paying all its "costs". First of all, this is indicative of the array of possibilities between free and non-free. It also allows clarifying the position of such companies in that regard and indicating some strategies that must be avoided from the viewpoint of free licences. We have observed that the role of Shared Source licences has diminished, due to the trust and popularity gained by truly-free software licences, and the criticism received thereby at the time.

The Sun Community Source License (SCSL) was an attempt to offer access to the code and programming environments of Sun Microsystems Inc., for instance Java or Jini, and to establish it as a standard. In this sense, it has had great success, especially in terms of Java. The "components" included in the Sun Community License were J2EE, the Java Developers Kit (JDK), Personal Java and Embedded Java, among others. The shared source era of Sun has nonetheless almost ended, inasmuch as in November 2006, Sun Microsystems released most of the programs forming the Java technological environment under the GPL (with the exception of Classpath).

The SCSL was, above all, a licence for developers. It is "open" mainly for research and development purposes, but allows Sun to maintain a strong control of the evolution of the program and programming environments. Conceptually, it was a licence that was halfway between the MPL and a non-free licence: it allows corrections, modifications and extensions, but any of these must be returned to Sun.

In 2006-2007, pressured by the free community, the rise of new free projects to create Java technologies to replace Sun software and the acceptance by Sun of the benefits of free software, the company began to adopt a position more favouring free software. It first opened Opensolaris, its operating system, under the CDDL and created a project and a community around the software. It later published its part of the patent pledge against Opensolaris users. Finally, in November 2006, it released its Java technologies under a GPLv2, with the Classpath exception (which allows using the libraries without a copyleft effect).

4.4.2. Microsoft Shared Source Initiative (MSSI)

Microsoft also created a series of over ten "semi-free" licences for part of its programs. They applied to the CE operating system for portable devices, CLI (Common Language Infrastructure) and the specifications of C#, and also included elements of Windows 2000 and XP. This "gesture" especially allowed the academic study of the technologies in question and, for commercial companies creating products running on such platforms, a better integration of

MSSI

About the MSSI, see the Microsoft website.

their programs with those of Microsoft. It also allowed Microsoft to disclose the source code of several applications to government organisations, under very strict secrecy conditions.

There were several types of licence as part of its Shared Source initiative. The basic model, for instance, the Shared Source licence of CE, opened the code to researchers and students: the source code could be downloaded and studied, and code modifications could only be used, modified and distributed for non commercial use, provided the same licence was maintained. Later, with the Windows CE Shared Source Premium Licensing Program, manufacturers of OEM devices had access to the source code of Windows CE and the right to modify and distribute the modifications commercially. They were nonetheless required to license any modification to Microsoft free of charge, allowing for it to incorporate such modifications in subsequent versions of the software after a six-month period.

Other MSSI licences contain variations of these rights granted and reserved. The licence for ASP.net, for instance, allows any commercial and non commercial use, but prohibits combining and distributing the ASP.net program with any free programs and especially under copyleft conditions.

In October 2005, Microsoft reduced its Shared Source licences to five: three basic licences and two variants, limited to the Windows platforms. The three basic licences are:

- **Microsoft Public License (Ms-PL)**. This is a permissive licence, copyleft for distributions made in source code format, but permissive for distributions in binary format. Contains a variation limited to technologies for Windows. Approved by the OSI and compatible with the GPLv3.

- **Microsoft Reciprocal License (Ms-CL)**. This is a reciprocal licence or copyleft, with effects similar to that of the Mozilla licence: the copyleft effect is defined based on the original files and the files of a "greater work" (using the original files) may be distributed under any licence. It also has a variant limited to technologies for Windows. It is approved by the OSI but not compatible with the GPL.

- **Microsoft Reference License (Ms-RL)**. This is a licence similar to the former Shared Source licences, which allows copying the program for internal use, but not its modification or distribution.

4.5. Free documentation licences

Free licences are applied mostly, but not exclusively, to software. A series of free licences have been created for documentation, especially as software is accompanied by technical documentation, which is often necessary for its use. It would not make sense to distribute free software without distributing the relevant documentation under similar terms. This led the FSF to create

the General Free Document License to accompany its programs. Additionally, following the trend to open knowledge, other licences have been created on documentation and materials, especially academic. We shall present an example: the Creative Commons initiative.

4.5.1. The GNU Free Documentation License (GFDL)

The GFDL is generally used for licensing technical documentation, user manuals and other relevant texts for free software. It is modelled upon the GPL, but changes its conditions to adapt to written text rather than software. The licence seeks balance to allow modifications (especially those necessary to document a modification of the software), maintain the authorship of the initial work and respect the ideas and opinions of the original authors.

The licence defines several elements of a document to establish the rights and obligations corresponding to each, for instance "secondary sections" (legal notices, dedications, acknowledgements, etc.) and "invariable sections" (secondary sections that cannot be modified).

The GFDL grants several rights relating to copying, distribution, modification, aggregation and combination, collection and translation of the original document. These rights generally granted, subject to the respecting the original authorship, maintaining certain identified parts of the text unchanged, and supplying access to a "transparent" version of the document (the equivalent of the source code of a program, being a legible copy, modifiable by a third party using free or generic programs, such as ASCII, XML with public DTD, HTML formats, etc.,).

Transformation of the text gives rise to a series of obligations: any derived work must change the title on the cover, indicate the original authors and any modifications, indicate where the original version may be found and maintain copyright notices and the licence. Additionally, certain defined sections must be maintained and the tone and general content of the secondary sections must remain unaltered. Any indication of endorsements must be eliminated from derived works.

As the GPL, the GFDL maintains the copyleft of the documents: any modification must be distributed under the same licence and cannot be combined with text from work under a more restrictive licence.

The licence not only applies to technical documentation for software. It may also be used for any text, specifically any "literary" work developed as free software: in collaborative works. In fact, Wikipedia (at www.wikipedia.org) is published under the GFDL.

In 2008, a new minor release of the licence was published, version 1.3, so as to achieve compatibility with the Creative Commons BY-SA 3.0 licence which we comment below. This is mainly so that wikis such as Wikipedia can use content under this CC licence in the wiki.

The GFDL is not the only free documentation licence. In part due to the controversy in relation thereto, many free software projects have created their own licences: the FreeBSD Documentation License, the Apple Common Documentation License or the Open Publication License, and the OR Magazine License (by O'Reilly).

4.5.2. The Creative Commons initiative

The Creative Commons initiative, often abbreviated to CC, is a project of Stanford University, in California, created by a series of copyrights experts, including Prof. Lawrence Lessig. It seeks to aid authors and creators to freely distribute their works for use by the public, thus increasing the number of creative works available to all. It is especially directed to literary and artistic creations and not software, and expressly recommends the GFDL for any computer documentation. Additionally, the CC proposes a private system, under United States law, to limit the duration of copyright protection to fourteen years, rather than the term agreed by law (generally, the life of the author plus seventy years) based on a public statement. Finally, it allows dedicating works to the public domain, also under conditions of United States copyright.

> Some rights reserved. The Creative Commons[9] initiative operates under a slogan that is a play on words on the regular copyright reserve of "all rights reserved". The slogan is "Some rights reserved", similar to that of the FSF, which is "All rights reversed". The freest CC licence would even allow including the expression "No rights reserved".
>
> (9)The Creative Commons project may be found at creativecommons.org.

In addition to a generic version of the licence, which is sought to conform to the international conventions on author's rights, there are versions adapted to the legal framework of each country: Spain, Peru, England, Japan, etc. (and linguistic versions, in Catalan, for instance). The latest generic version, 3.0, contains a compatibility agreement to allow the equivalence of licences between these "local" versions.

The strategy of the CC has been to create a series of modular licences establishing what rights are granted to the licensees.

The licences contain a core of terms that are common to all variants and then particularities regarding the grant of rights. The core elements include:

- Notices of authorship and copyright are required to be maintained ("BY").
- Internet links may be established in works published on such medium.
- No modifications to the licence are permitted.

- No technological mediums may be used to restrict the legitimate use of the work (in other words, no DRM technologies).
- They apply in all countries of the world.
- They are irrevocable and have a duration equal to the term of the copyright protection.
- They offer a warranty of ownership and non violation of third-party rights (to increase confidence in the reuse and redistribution of the work).
- The author or owner of rights is allowed to distribute the work under a different licence.
- They contain a special exception allowing P2P file-sharing, which is not considered a commercial activity, provided it is not for profit.

Regarding the grant of rights, authors may choose the rights that are reserved and granted in the licence based on three criteria:

- Commercial use ("NC", for non commercial use restriction).
- Allowing derivative works or not ("ND" – no derivatives).
- Reciprocity or copyleft ("SA", for share alike).

The website www.creativecommons.org also contains an automated tool for creating the licence based on the answers to questions on such criteria. A CC licence is proposed to users upon the basis of two questions:

- Whether or not to permit commercial use:
  - Commercial. Allows any type of use, including commercial.
  - Non commercial (NC). Allows any type of exploitation and derivation, provided it is for non commercial purposes.

- Whether or not to allow the creation of derived works:
  - No derived works (ND). Modifications are not allowed for the creation of derived works.
  - Share alike (SA). If derivative works are allowed, then redistribution of the work and derived works must be solely upon equal terms as the original licence (copyleft).

Thus the most basic and permissive licence is the Attribution license (BY), which merely requires that credit be given where due and allows everything else, similar to a BSD style free software licence: no commercial restriction, no share alike.

The Attribution-NonCommercial-ShareAlike (BY-NC-SA) allows modification, requires maintaining the same licence in derived works and prohibits commercial use. The MIT OpenCourseWare licence is of this type. Another text with this licence is "HOWTO: Installing Web Services with [free software]".

The Attribution-NonCommercial licence requires that credit be given where due and restricts commercial uses. The Electronic Freedom Foundation, at www.eff.org, uses this licence.

The tool creates and offers the user the text of the licence. Licences come in three formats:

- An easy to read version: a very easily understandable summary ("Commons deed" or "Human code"), with icons, which we shall mention hereunder.
- A legal version for lawyers: the complete version of the licence ("Legal code").
- A machine-readable version: an expression in RDF and XML metadata so that an automated computer process may understand the licence in the context of the semantic web ("Digital code").

4.5.3. Freeware and shareware licences

We only wish to stress here that shareware and freeware licences are not free software licences. Although the relevant programs may be distributed free of charge, they do not provide access to the source code and, in their majority, they do not respect the minimum conditions for being free or open: the four basics freedoms of the FSF or the OSD definition.

5. Free software licences in practice

After the previous analysis of free software licences, this section starts with a comment on and clarification of certain myths or misconceptions with respect to various legal aspects of free software. We then comment on several key issues related to free software licensing, including how to choose a free licence, the issues raised by contributions to free software projects, compatibility between licences, and other topics.

5.1. Some legal myths about free software ... to debunk

Over the years certain myths or misconceptions have arisen with respect to various legal aspects of free software, not least due to FUD (Fear Uncertainty and Doubt) spread by those who do not necessarily agree with the tenets of the free software movement. Here, we comment on these misconceptions and try to determine the truth of fallacy behind them.

> There are other myths relating to the technological or commercial aspects of free software that we shall not address here: lack of support and maintenance, lack of security, risk of forking, the possibility of introducing damaging elements in free software, lack of viable business models based on free software, etc.

5.1.1. Copyleft goes against author's rights

This myth is based on the belief that copyleft (and free licences in general) creates a new intellectual property legal framework: *copyleft* "rather than" *copyright*.

Quite the opposite, as seen above, free software licences are based directly on the current author's rights or copyright law, whether it be author's rights under continental style or the copyright of English-speaking countries. The authors of free software use the rights established by this legal framework (exclusive rights to exploit and/or authorise the exploitation of their work) to grant the licensees the non-exclusive rights established in the free licences and defend these rights from infringement.

> In MySQL AB vs Progress Software, MySQL AB defended its ownership of rights in the database application MySQL. It initiated proceedings against Progress Software for the violation of author's rights and of the licence terms of the GPL to the MySQL program.
>
> In Germany, the several courts have now decided in favour of the rightsholders in the netfilter/iptables project, in relation to infringement of the terms of the GPL based on authors' rights / copyright law. They could thus enforce the licence obligations on a licensee who had breached the terms of the licence and (for distribution without the source code and without a copy of the licence) thus potentially infringed their copyright.

Let's consider, for instance, two important characteristics of free software: freedom of use and copyleft conditions.

- Regarding the first, the legal framework allows the owners of work to define the scope of the exploitation rights granted to third parties. Rather than restricting the uses (as is done by most non-free licences), a free software licence permits them to the maximum permitted by law. This does not go against author's rights, but rather is an exercise thereof.
- Regarding copyleft, a developer may create and distribute a derivative work of free software as permitted, under certain conditions, by the owner of the original work on which it is based. If such conditions – for instance, to distribute the derivative work under the same licence (copyleft)– are not met, the original licence shall be cancelled and the distribution of the derived work shall constitute a breach of the copyright rights of the licensor. The copyleft acts legally as a termination clause.

In the USA, in Jacobsen v. Katzer the US Court of Appeals held that Artistic licence terms were enforceable conditions on the permission to exercise copyright rights granted in the licence, and therefore a licensee in breach of the licence would be in breach of copyright, the licence being revoked.

Therefore, there is no contradiction or opposition between legislated author's rights and the rights granted or reserved under a free licence. Moreover, it could be argued that, given that a free software licence respects the exceptions and uses permitted to the user under our legal framework, it conforms better to the law than many non-free licences.

E. Moglen, "Enforcing the GNU GPL", Linux User, 12/08/2001

"The GPL, on the other hand, subtracts from copyright rather than adding to it [user restrictions, for instance]... Copyright grants publishers power to forbid users to exercise rights to copy, modify, and distribute that we believe all users should have; the GPL thus relaxes almost all the restrictions of the intellectual property system".

Enforcing the GNU GPL

Article to be read online at E. Moglen's site.

5.1.2. Free software has no owners

There is nothing further from the truth, from a legal viewpoint. The author's rights/copyright legal framework automatically grants author's rights to the creators of software. And the sole obligation –or almost sole obligation– common to all free licences is to maintain the notices of the rights of the initial creators of the software (the famous "copyright notice"). There is therefore always an owner of the rights to the software and, in the case of free software, ownership is clearly indicated in the files.

5.1.3. Free licences compel authors to assign their rights

With the exception of moral rights, which are non-transferable, author's rights may be assigned or licensed, but solely with the express consent of the owner. Free licences are "non exclusive" and cannot "strip" the ownership of the software from their creators. Free licences subject to copyleft do compel licensees to use the same licence (non exclusive) in any future distribution of modifi-

cations or work derived from the original software with these licences and to publish the relevant source code, as a condition of the right to redistribute the modification, but do not force them to "assign the software" (or their rights thereupon) to anyone.

5.1.4. Free software cannot be subject to commercial use

Another misconception: as we have seen, there are no limitations to the use of free software (freedom 0); the only conditions imposed, sometimes, refer to its subsequent modification and distribution. Free licences do not affect the end users.

5.1.5. Free software and non-free software are incompatible

Another myth is that free software is incompatible with non-free software if they are run on the same computer system or platform. If this were true, no non-free application, such as the Oracle databases, could be run on GNU/ Linux, OpenBSD or the web Apache servers. And vice-versa, free applications such as MySQL could not be run on non-free operating systems such as Oracle's Solaris or IBM's AIX. What may give rise to incompatibilities is the integration or mixing of copyleft software and non-free software, as we shall discuss hereafter.

5.1.6. Free software cannot be integrated or mixed with non-free software

This claim holds that free software, in general, cannot be mixed or integrated with non-free software in the same application without affecting it and, accordingly, without breaching its conditions of user. A stronger way of expressing this is claiming that free software and GPL software in particular is viral and "infects" other applications: any application integrating GPL software becomes GPL software. This statement is partially untrue.

- Integration by end user. Free licences do not restrict the use of software with other applications: the possibility of its modification is a condition of its being free and there are no restrictions on its use. It is therefore necessary to distribute the source code with the object code or to make it available to the recipient. Nonetheless, any integration of free software (permitted by the free licence) with non-free software may be considered a modification of the integrated non-free software (if the source code is available to make it). Depending on the restrictions contained in the non-free licence, such modification could constitute a breach, regardless of whether the integrated program is free, non-free or redistributed. This is not a problem of the free software, but of the non-free software licence.

- Integration by an intermediary. Where restrictions may exist in relation to the integration of software of various types, whether free or non-free,

is with respect to its subsequent distribution. Permissive licences allow mixing and redistributing their software with non-free licences. On the other hand, *copyleft* licences prohibit redistribution with non-free licences of a "mix" of software with these non-free software licences, which practice has come to be known as the privatisation of free software. Certain free licences contain clauses seeking to partially allow this integration, such as the LGPL or the MPL, which we have discussed previously.

5.1.7. All free software is licensed in the same manner (upon the terms of the GPL)

There are substantial variations between the more than seventy free and open source software licences recognised by the OSI. When discussing licences, it is important to be much more careful in the use of the term free software, and distinguish between free licences in general, licences subject to copyleft and licences that are neither free nor open. It is important to clearly understand the terms open source, persistence or reciprocity and copyleft, which are characteristic of such free licences.

5.1.8. Free licences require the publication of modifications to the code

This is one of the most incorrect ideas propagated in respect of the workings of free licences. We shall distinguish between the position of end users and intermediaries (developers of programs for third parties):

- **End users**. Most free licences do not require that users should distribute their modifications or adaptations of free software (derived works, in legal jargon) or should publish them or contribute them to the development of the modified application. Some licences do require the latter, in some particular cases, solely in relation to corrections or modifications of the central code or kernel of the program. As we shall see, these obligations do not apply to additional elements added to the kernel or any extension of the application. Therefore, the end user shall not be required to publish their works based on free software.

- **Professionals and companies developing programs**. Those developing programs for clients are not required to distribute to the public (or to the original authors) any modifications to free software. What they are required to do is respect the original free licences, many of which require providing the source code to users or clients receiving them or, if only the object code is distributed, offering the source code to any third party (the GPLv2) or the recipient (the MPL, GPLv3) for a certain period. This is one of the requirements for using free software subject to copyleft.

Supplementary content

The Apple Public License 1.x required that any modification of the original program be sent to Apple and this was one of the reasons for it not being considered a free licence.

5.1.9. With free software there are no liabilities or warranties

It is necessary to recognise that this may be true, under current free software licences, especially when the software is distributed free of charge. Nonetheless, there are legal doubts in terms of the effectiveness of warranty disclaimers and liability limitation clauses, which may not be valid with respect to consumers, at least.

The myth, in reality, consists of thinking that non-free licences give greater warranties and accept a higher level of liability. Many non-free licences seek to limit the liability of the licensor (author or distributor) in terms quite similar to free software licences. Indeed, they usually seek to limit contractual warranties, for instance, to the repayment of the purchase price in case of a fault with the software is identified within a limit of ninety days.

Another argument regarding warranties and liability is that with virtual distribution systems over the internet, it is difficult to identify licensors and thereby claim any compensation. Many sites distributing free software, such as Sourceforge, are not the owners – licensors, or even "official" distributors similar to those who distribute proprietary packages.

Nonetheless, in some cases, such as that of the FSF or in businesses based on the distribution of free software packages such as Red Hat or Suse (Novell), there is an identifiable legal entity that could be subject to an action for liabilities if necessary. Furthermore, the obligation to maintain the copyright notice allows rightsholders of any component that could prove defective to be identified, even if they are not necessarily who distributed the program to the affected party.

In addition, free licences allow free software distributors to add warranty clauses (with or without an economic consideration), which is done with many packages destined for commercial distribution.

5.2. Some legal issues relating to the licences

After going over the myths surrounding the legal effects of free software licences, this section is intended to provide some practical comments on the legal issues of free software and free software licensing.

Besides technical and economic aspects, there are a variety of important legal issues to be considered so as to ensure the success of any activity involving free software, whether it be its creation and distribution or its implementation in public or private organisations, and it is highly recommended to establish the appropriate legal strategies.

Understanding broader legal issues relating to free software, such as the legal consequences of inbound licences or interrelations between various concepts we have discussed here (copyleft, compatibility, licensing regime, etc.), should help us to manage free software projects better and reduce perceived difficulties and FUD.

The subjects that we shall address in this section, the "practical effects" of free software licences, relate especially to the management of intellectual and industrial property in free software based projects. We will specifically comment on:

- How to choose a free licence.
- How to manage the contributions to free software projects.
- Compatibility between licences.
- Dual or multiple licensing.
- The effect of licences on free software forking.

5.2.1. Choosing a free licence

The terms of a free software licence to be applied to a project normally result from a compromise between several objectives, determined by the authors or team leaders (coordinators) of the project in question. Generally speaking, the following objectives are considered, which may to a certain extent conflict with one another:

- Guaranteeing certain basic freedoms common to all free software (use, copy, modification, redistribution, patents, etc.).
- Imposing some conditions or restrictions (recognition of authorship, absence of warranty, use of trademarks, etc.).
- Procuring that the modifications and derived works should also be free, or not.
- Reserving some rights.
- Maintaining control over the evolution of the program.

Each project therefore has its objectives and criteria in terms of the licence.

In general when choosing a license for a project, it is recommended to use an already-existing licence rather than writing a new one. This issue has become increasingly more important due to the proliferation of free software licences (to the point that the OSI is attempting to reduce the number of certified licences). A general trend is to rely on one of the more common licences: GPL, LGPL, BSD, MIT, MPL, Apache, CPL, etc. This offers the advantage of increasing compatibility probabilities between programs and components. Another possibility is to look to a "third generation" licence, such as the GPLv3, OSL 3.0 or EUPL 1.1 licences (copyleft/reciprocal), or the Apache 2.0 or AFL 3.0 li-

cences (permissive), which cover certain issues that have arisen recently, such as patents, trademarks, remote use over a network, etc. and, for our purposes, may be better suited to the European legal framework.

Many free software exponents recommend the use of a licence compatible with GPL, especially as it is used by almost 75% of the free software projects (not necessarily 75% of the available free software), but also as GPL generally receives more support from the free development community. There is some controversy in this regard, inasmuch as there are projects and developers that refuse to accept GPL code and others solely accept code under GPL or a compatible licence.

The main criterion is usually whether a project wishes to impose copyleft or reciprocity obligations: the obligation for developments based on the original software, generally derived works, to maintain the same licence for redistribution. The GPL, for instance, seeks to enlarge the pool of free software available and maximise the freedom of end users: it therefore includes the copyleft clause. The GPL also has the practical effect of limiting forking (a separate evolution of the same software over various projects – see below). As we have seen, other licences such as the LGPL or MPL have a weaker copyleft effect, applied solely to the original work (or component) and to any specific modification. It does not extend to applications "integrating" or "using" the free component. These allow for the integration or linking of the original components with another code, to create what are known as "larger works".

Some questions that are often considered include:

- Do I wish to allow the privatisation of derived works and modifications?
- Do I want the developers to return their modifications to the free community in general, or to me, the initial author, in particular?
- Do I want to allow the licensees to merge or link their program to mine?
- Do I want a greater dissemination of the program and to attempt to establish a standard?
- Do I want to obtain licence fees from my program, based on its use (commercial or otherwise), while at the same time permitting free development?
- Do I have a unique innovative program, or is it just another content management system, for instance, when there are already many of them available, both free and non-free?
- Do I have obligations with third parties in relation to the code incorporated in my program?
- Does my program need to be run with any other program in particular? Are there restrictions thereon?
- Do I want to encourage other developers to participate in my project and contribute code or test hours?
- Has my application been designed to be embedded into a device, along with other, non-free, software?
- Is there a "predominant" licence in the sector of my particular software (for instance, a language or libraries)?
- Is there any risk for anyone holding or applying for a patent on an element or an aspect of the program?

The following chart, which is already a "classic" and appears in almost all documents on the subject, takes into account the main free licences and assists in the selection of a licence.

Criterion Licence	Allows linking with non-free programs	Can be mixed with other software	Allows derivative works to be non-free	Grants a patent licence
GPLv2/v3				Yes
LGPLv2/v3	Yes			Yes
MPL	Yes	Yes		Yes
BSD	Yes	Yes	Yes	(no)

Another option lies in the choice of a dual licence policy, which we shall discuss below. This system, in which the program is distributed with different licences (normally one copyleft, the other restrictive licence), allows income to be obtained based on the non-free version of the software, and collaborating with a "community" to improve the free program on the free version.

Supplementary content

This is the licensing system used by MySQL and Trolltech/ Qt, among other companies, see below.

Additionally, if the program is modular, it is possible to use different licences for different components, provided they are compatible in relation to the communication mechanism used by the components (i.e. depending on the degree of integration).

Another strategy for client/server systems is the use of a free licence for the client and a non-free licence for the server.

Suggested reading

- Zooko O'Whielacronx: Quick Reference For Choosing a Free Software License http://zooko.com/license_quick_ref.html.
- Bruce Perens: The Open Source Definition, en "OPEN SOURCES", p185.
- Donald K. Rosenberg: Evaluation of Public Software Licenses, online at http://www.stromian.com/Public_Licenses.html (last visited March 29, 2001).
- Frank Hecker: Setting Up Shop: The Business of Open Source Software, online at http://www.hecker.org/writings/setting-up-shop.html.
- Mike Perry: Open Source Licenses, online at http://fscked.org/writings/OpenSource.html.
- Brian Behlendorf: Open Source as Business Strategy, en "OPEN SOURCES".
- Rex Brooks: Open Source Licenses Overview, online at http://www.vrml.org/TaskGroups/vrmlipr/open_source_overview.html.
- Eric Kidd: A History of "Open Source", online at http://discuss.userland.com/msgReader$19844#19889 (Aug. 19, 2000).
- Estudio POSS / IDA (Unysis para la Unión Europea), pp60-65.
- The Mitre Corporation: Use of Free and Open-Source Software (FOSS) in the U.S. Department of Defense, Version 1.2, 28 Octubre 2002, p. 15.

5.2.2. Licences for contributions and authorship

An essential element that must be taken into consideration when managing the legal issues of a free project is that of contributions to the project. The history of Netscape shows the difficulties that one could face if required to choose to free one's software or change from one free licence to another.

The problems that could arise when accepting third party contributions include:

- Software obtained from an unsafe source (may have been copied).
- Software contributed with another licence (an incompatible licence).
- Software covered by pre-existing obligations, either by third-party licences or commitments binding the author-licensor (the problem faced by Netscape).
- Patents granted on an element of the code.

Supplementary content

For those in charge of the project, it is necessary to follow carefully the code that is contributed and keep a log of authorship and versions, identifying each software component.

Contributors tend to contribute code under various legal instruments: the project licence, a licence compatible with that of the project or a more individualised assignment (an agreement on contributions). There is a debate about whether contribution agreements should be mere permissive licences or joint or full assignment.

For those in charge of the project, it is important to have sufficient rights to incorporate and distribute the contribution in the project code. On top of this, they may wish to have wide rights so as to be able to change the licence in the future (e.g. evolve from GPLv2 to GPLv3), and finally, take legal action to defend against breaches of IP rights.

Accordingly, there are projects that request contributors to grant a complete and exclusive licence to the entity coordinating the project (in Anglo-Saxon countries, an assignment) with warranties in respect of the ownership of the rights to the contribution and, eventually, a patent licence. This allows project coordinators to keep certain control over the outcome of a free project and protect themselves from the risk that the code have come from unsafe sources (for instance, copied from other software) or that any original author that has requested a patent on the software or, in the law of English-speaking countries, revoke the licence.

Supplementary content

The FSF usually demands that any programmer contributing more than ten lines of software to a GNU project should assign the code exclusively to the FSF.

Other projects are happy taking contributions under the project licence or a compatible licence (often a permissive licence such as a three clause BSD or MIT) or under a non-exclusive contribution licence agreement.

5.2.3. Compatibility between licences

We have raised the issue code and licence compatibility on several occasions.

A program is legally compatible with another if their codes may be mixed so as to create a derived work (made up of elements of each) and you can distribute the result of the integration without infringing upon the licences of both – i.e. the conditions of both licences may be met when redistributing the results.

To distribute software integrating several free software components legally, it is essential that the licences on the components (inbound licences) be compatible with the licence chosen for the distribution of the final product (outbound licence). For instance, as the new (or three clause) BSD licence allows almost any form of exploitation of the associated software, BSD software may be mixed or integrated with other programs with almost any licence (the GPL, for instance) and the result may be distributed under such licence without infringing upon the BSD.

Supplementary content

Several projects have strived to be compatible with the GPL, such as Python, Qt and Vim, and even Mozilla added an additional clause to permit dual licensing.

Licences with copyleft are incompatible among each other, except where specifically agreed (such as the EUPL, LGPL or AGPLv3, or multiple licensed MPL code). When redistributing a program integrating two components with different copyleft licences, the use of one of the licences for the distribution of the final product would constitute an infringement of the terms of the other.

Supplementary content

The FSF offers a complex chart on the compatibility between the GPLv2 and the GPLv3, covering several cases. See the FSF site.

Probably the greatest debate surrounding free software is the question of licence compatibility and linking: even if two programs cannot be integrated together as a whole (e.g. statically compiled together and linked to create the executable), could they be dynamically linked? This may be achieved in various forms, as we have discussed in relation to GPLv2: for instance, inserting an API between one component and another or creating dynamic links that activate when run.

The MPL expressly provides that applications under other licences may be linked with or use software under this licence, as a larger work. Usually, the non-free program must be linked to the software under MPL by means of an API. The API would generally be part of the original program, or a customised modification, and will therefore be subject to the MPL, while the linking program can be licensed under any licence, even non-free.

5.2.4. Dual or multiple licensing regimes

As free software licences are non exclusive, a program may be distributed by its rightsholder under two or more free licences, or under a free licence and a non-free licence, under a dual licensing system. A rightsholder is not restricted as to licensing the code, unless an exclusive licence has been granted or, in certain cases, confidentiality agreements have been established.

Several programs are distributed in this manner:

- MySQL is a database engine distributed under GPL for independent use and with a non-free licence for integration with commercial products.
- eZ-publish is an internet content management program that also has a dual GPL/non-free licence.
- The MPL 1.1. allows the owner to establish whether the program is distributed with the MPL or other licences (Clause 13).

From the viewpoint of the software rightsholder, the possibility of using a second licence allows him/her to offer different solutions to different stakeholders: community members (free licence) or clients (non-free licence or free

subject to restrictions). This also allows controlling the price, quality and liabilities with respect to the software on the commercial side, and, eventually, releasing it (as Sun has with the Java technologies).

To accomplish this strategy, in the case of using a non-free licence as a second licence, it is essential that the rightsholder ensures ownership of the rights in all components incorporated in the product, or at least that the licences on any third party components are permissive (BSD, MIT), thus permitting relicensing under many terms. Therefore, projects that dual license their products tend to centralise the author's rights in sub-components in their hands, as is done by companies such as MySQL AB and Trolltech.

> This requires community contributors to assign their rights to their contribution to the project, to ensure there is no parallel use of valuable contributions (or provide very wide contribution licence terms). Projects should also establish restrictive licences with commercial partners and control patent risk.

The success of the strategy depends on the likelihood of third parties (whether commercial or within the free software community) creating a similar product or fork, which would compete for the product with a non-free licence. The use of a copyleft licence for the free licence usually prevents third parties from taking the free program and privatising it for commercial purposes, competing with the commercial version with a non-free licence.

Another element for controlling software in the context of this strategy lies in trademarks, a legal tool that we have already discussed. Trademarks are used by the Apache Foundation with respect to its software (the web server, Tomcat, etc.), by Sun in relation to Java® and by several professional free software companies such as Sugar® (CRM), Compiere® and Openbravo® (ERP), Pentaho® (Business Intelligence), Alfresco® (document management) and Zimbra® (mail and groupware), Socialtext®, etc.

5.2.5. Free software licences and forking

The concept of forking or division comes from computer multitasking: it refers to the division of one task or process into two. For instance, a task may remain active while another is stopped. The division or forking of a program takes place when the software is modified and the modification is developed separately as a separate branch or project, under another coordination team, and is often distributed under another name and perhaps another licence. Examples of these are OpenBSD and NetBSD, divisions of the original Unix BSD, and Compiere and Adempiere, etc.

> MySQL defines forking as: "Forking [of MySQL] means to divide the source code of the MySQL database in a repository kept separately, so as any development of the original code requires a manual operation to be transferred to the forked software, or that the forked software begins to incorporate functions not present in the original software".
>
> The program that has suffered this phenomenon most is UNIX, leading to the creation of up to ten variants. Some UNIX variants were created because their original authors (AT&T and the University of California, Berkeley) distributed them under permissive free

licences, allowing new versions (even non-free) to be created: Unixware by Novell, Open Server SCO, Solaris by Oracle, AIX by IBM, etc. This has given rise to legal problems, for instance, in the original case between AT&T and the University of California, and more recently between SCO and IBM, and others.

The possibility of forking is relevant to developers, inasmuch as it provides an indication of the possible technical and legal or commercial evolution of the software. From a technical viewpoint, "forked" versions tend to be technically incompatible (or "non interoperable") with the original programs. From a legal and commercial viewpoint, these versions may compete with the original product and be distributed with a different licence, either free or commercial, fostering legal incompatibility.

There are several reasons for forking.

- Developers fork free software because they (legally) can: free licences allow the modification of free software and the redistribution of such modifications. For instance, the BSD licence allows derived works to be created from the original software and changing the licence on the modified code. The new program could start as a mere variant from the original (for instance, OpenBSD with regard to NetBSD) and then diverge more and more.
- Another reason lies in the management of the development equipment and in disagreements between the programmers or owners of the software (Mambo/Joomla, is an example). For instance, if the need for an extension or new module is identified and the coordinator does not agree, it is quite likely that someone may create a forked version to integrate such a module.

Free licences have a direct influence on forking:

- Generally speaking, free software licences enable forks: software subject to a non-free licence or shareware cannot be forked.
- Software licensed under terms that do not allow commercial use and/or demanding the return of the modifications to the original author cannot be forked due to the centralising control exercised by the author (for instance, Ghostscript and the Aladdin licence).
- Software under strong copyleft licences such as the GPL tend not to be forked: derived works must be made available to licensees under the terms of the same licence, and the original project may, therefore, usually have access to the derived code so as to reincorporate any improvement and make it part of its own version.
- Software with weak copyleft licences, such as the Artistic, the MPL or the LGPL may be forked more easily, as they allow the creation of non-free or free variants by "aggregation".
- Software under a permissive licence (BSD or similar) can and is easily forked, as it allows binary distributions without the obligation to publish

the source code, and therefore, the original project may not have access to the changes made in the fork.

6. Conclusion

In this module, we have looked at software licensing in general and at the legal aspects of free software licences and licensing in particular.

There are currently "many" free licences and free licensing models. In fact, there are so many recognised licences (some seventy have been approved by the OSI as "open sourced") that the OSI has established a committee against licence proliferation, something that we comment upon here.

On its website, the OSI has classified (arbitrarily, according to some) the available licences as "most popular", "special purpose", and "other" and "redundant". We have observed with interest on Sourceforge, the largest repository of free software, the evolution of the use of the various licences, with the GPLv2 still in the lead, accounting for some 65% of the projects (which does not necessarily imply 70% of the code).

However, software and licences are not created and used in a stable and invariable environment, but rather are immersed in a changing medium, evolving quickly, both technically and legally. For instance, dynamic linking and interpreted languages did not really exist when the GPL v.2.0 was drafted, nor was there legislation for the protection of rights management information and technological protection measures. Consequently, to maintain software freedom, both technology and the legal framework need to evolve together, e.g. by adapting licences to new technological developments (SaaS, web-services) and legal developments (DRM).

Supplementary content

We have seen that the ¡GPLv3 was drafted taking into account both technology and legal change, and the Mozilla project has just announced (March 2010) that it is starting a community process to review the MPL 1.1.

Also, we wish to stress the importance of understanding software licences, their choice and how to comply with their terms, in relation to the use, distribution and marketing of free software or products based on free software in increasingly-complex development and production environments, such as distributed environments (web-services, software as a service, etc.):

- For software developers it is also fundamental to manage their own intellectual property and that of the contributors of code to the projects they run.
- For users, it is important to understand the licences applied to the programs they use, the rights they enjoy and the obligations they must comply with, and the differences and compatibilities between licences.

Regarding the professional life of the student, we think that it is important that they be able to assess the commercial and technological consequences of the legal issues that we have described and commented on here: what may be done in relation to the legal effectiveness or ineffectiveness of licences, how to take advantage of the possibilities of multiple licences, what documentation

or checklists could be useful for IP management, what strategy or tactic must be adopted when facing legal doubts with respect to free software, and which decision process is most appropriate.