LES LIVRETS BLEUS DU

LOGICIEL LIBRE

Fondamentaux juridiques

Collaboration industrielle et innovation ouverte

2e édition

PATRICK MOREAU
CAMILLE MOULIN
JEREMY PAPPALARDO
FRANÇOIS PELLEGRINI

Préface de
STÉFANE FERMIGIER

Aquinetic
Systematic
Paris Region Digital Ecosystem

LOGICIEL LIBRE

Remerciements chaleureux aux auteurs de cet ouvrage :

Patrick Moreau,
Fondateur de TILC, responsable
des partenariats industriels du CNRS

Camille Moulin,
Consultant chez Inno3

Jeremy Pappalardo,
Juriste à l'École Polytechnique

François Pellegrini,
Président du cluster Aquinetic,
Professeur à l'université de Bordeaux, chercheur Inria / LaBRI

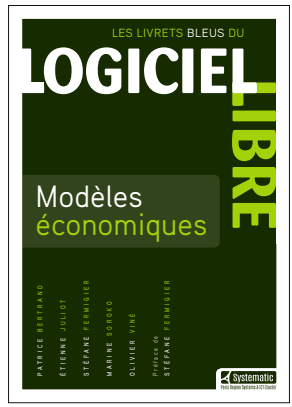
Préface de **Stéfane Fermigier**, président du GTLL, fondateur d'Abilian

Une première version de ce livret bleu a été éditée en juin 2016. Cette deuxième version intègre de nombreuses corrections, issues notamment des commentaires de **Roberto Di Cosmo**, **Sébastien Dinot**, **Gilles Lehmann** et **Simon Tournier**, que les auteurs remercient vivement.

*Cet ouvrage a été édité en coopération avec le cluster partenaire **Aquinetic**.*

Remerciements à Muriel Shan Sei Fan pour le travail d'édition, à Didier Méresse et Mickaël De Clippeleir (Nord Compo) pour la conception et composition et à Romain Berrendoner pour sa contribution très tôt à la conception de cet ouvrage.

LES LIVRETS BLEUS DU GTLL : DES REPÈRES POUR COMPRENDRE



© GTLL, Systematic Paris-Region, 2017 – ISBN 978-2-9549444-8-7
Diffusion sous licence CC BY-SA-NC-ND 4.0. *Nous contacter si vous souhaitez rediffuser ce document dans un cadre commercial.*

Sommaire

Chapitre 2
Cadre juridique spécifique du logiciel libre 16

Préface

Par logiciel libre – et open source, l'insistance sur la distinction ne faisant parfois qu'opacifier le débat – on entend bien des choses selon les contextes et les personnes qui s'y intéressent. Pour certains, c'est un gisement (inépuisable ?) de « briques » logicielles qui accélèrent en entreprise le développement d'applications modernes ou de systèmes complexes ; c'est aussi un mode de collaboration qui a prouvé son efficacité dans le monde du logiciel et qui s'étend depuis dix ans à d'autres secteurs de l'économie et de la société ; c'est un accélérateur de l'émergence et de la diffusion des nouvelles technologies ; c'est encore le fondement de modèles économiques qui ont bousculé le secteur du logiciel. Pour d'autres (et souvent les mêmes), c'est enfin un combat philosophique, voire politique, qui vise à rééquilibrer les pouvoirs entre les créateurs et les utilisateurs d'une technologie omniprésente, voire intrusive.

Quelle est donc la place du droit dans un débat qui semble principalement centré sur la technologie, l'organisation du travail, l'économie et la philosophie ?

À l'origine, ce sont des valeurs – liberté, partage, communauté – qui fondent le logiciel libre. Celles-ci, pour être actionnables, doivent être **traduites en droit**, et ont donné lieu à des principes – les fameuses définitions de la FSF et de l'OSI – puis enfin à des **licences** qui traduisent concrètement et dans un cadre juridique préexistant les intentions des producteurs de logiciel libre. Dans ce contexte, on parle tout autant des *utilisateurs finaux*, que des organisations ou des personnes qui *réutilisent* les logiciels libres sous forme de composants pour créer de nouveaux logiciels, des systèmes complexes ou une offre de service. En posant clairement les droits et devoirs des utilisateurs, les licences de logiciels libres **clarifient et sécurisent** les relations entre les membres d'écosystèmes de contribution extrêmement larges, et posent la question de l'équilibre du partage de la valeur.

Depuis plus de 30 ans que les premières licences libres existent, leur domaine d'application s'est considérablement élargi. **Confrontées aux nouvelles questions émergeant de l'évolution des technologies** (APIs, Web, Cloud, Internet des Objets…), **du droit** (voir les débats aux États-

Unis sur le copyright sur les interfaces utilisateurs ou sur les brevets logiciels), ou **de l'économie** (nouveaux modèles d'affaires fondés sur le logiciel libre), ces licences ont évolué, et de nouvelles licences sont apparues. À tel point qu'aujourd'hui, leur nombre peut dérouter de prime abord.

Heureusement, elles peuvent facilement être classées en quelques catégories, et celles qui sont d'usage courant sont en nombre relativement restreint – une dizaine. L'application d'un petit nombre de principes, exposés succinctement dans ce livre, permet aux dirigeants mais aussi aux développeurs, chefs de projets et architectes techniques de **maîtriser la complexité inhérente à l'assemblage de centaines de composants open source** dans le cadre des projets dont ils ont la charge. Les producteurs (aussi appelés *éditeurs*) de logiciel libre trouveront quant à eux des éléments de décision relatifs au choix des licences de leurs logiciels, **en fonction des modes de collaboration et de partage de la valeur qu'ils souhaitent privilégier en aval**.

Rappelons en regard que dans le monde du logiciel privatif, encore appelé logiciel propriétaire, il se trouve autant de licences que d'éditeurs – donc beaucoup plus que de licences libres –,

que ces licences imposent, pour la plupart, beaucoup plus de contraintes, et qu'elles ne sont pas guidées par les mêmes principes favorables à la réutilisation et à la collaboration, et à la **croissance organique d'un patrimoine logiciel**.

L'utilisation – dans tous les sens du terme – de logiciels libres, dès lors qu'elle est maîtrisée, en plus de tous les autres avantages qu'on leur connaît – **innovation accélérée**, **interopérabilité**, **qualité** –, offre aussi un avantage indéniable en termes de **simplification juridique pour les entreprises** – et donc de maîtrise du risque.

Stéfane Fermigier
Président du Groupe thématique Logiciel Libre
du pôle Systematic Paris-Region

Réfléchir sur le logiciel libre, c'est réfléchir en termes de valeur et de « valorisation », et définir les droits et obligations qui encadrent la répartition de cette valeur. Nous tentons ici de rafraîchir ces notions et de poser les bases d'une compréhension du logiciel libre en tant que propriété juridique, qui permette justement de revisiter la conception de valeur attachée.

Que l'on dirige une entreprise, que l'on soit chercheur, ingénieur, CEO, CTO, chef de projet, ou encore développeur, on ne peut s'abstraire de la question de la valorisation du logiciel. Le présent ouvrage complète celui consacré aux Modèles économiques du logiciel libre édité par le GTLL du pôle Systematic Paris-Region.

Après un rappel du cadre juridique du logiciel, nous aborderons les spécificités du logiciel libre, puis détaillerons les droits et obligations en cas d'usage ou d'exploitation de logiciels sous licence libre.

Chapitre 1

Cadre juridique générique du logiciel

Les règles communes applicables à tout logiciel, libre ou pas, méritent d'être rappelées.

La protection du logiciel

Logiciel et droit d'auteur

Lorsque nous utilisons le mot « logiciel », nous faisons en réalité référence à deux éléments :

- **le code** : c'est-à-dire – en fonction des langages de programmation – le code source ou le script, ainsi que toutes traductions automatiques qui en découlent, telles que le code objet ou le code exécutable ;
- **le matériel de conception préparatoire** : c'est-à-dire l'ensemble des fichiers et documents qui contiennent « en germe les développements ultérieurs » : cahiers de spécifications, description des algorithmes implémentés, logigrammes, etc.

Le logiciel est protégé par le **droit d'auteur adapté au logiciel**. Il s'agit en fait quasiment du même droit que le droit d'auteur classique qui protège le poète, le sculpteur ou l'architecte. Cela

n'est pas si étrange : le code source d'un logiciel est en définitive un texte rédigé dans un langage spécifique (le langage de programmation), lequel a ses propres règles grammaticales et syntaxiques.

Pour qu'un fragment de code source reçoive la protection du droit d'auteur, il faut qu'il puisse être considéré comme «**original**», c'est-à-dire qu'il reflète la personnalité de l'auteur qui l'a écrit. Nul besoin que les idées qu'il exprime soient nouvelles ou particulièrement innovantes. Écrire un nouveau navigateur web, c'est créer une œuvre originale, même si les idées et principes ne sont pas nouveaux, tout comme pour un énième roman de mousquetaires.

Une protection sans titre

Un principe important en matière de droit d'auteur est que la protection naît **«du simple fait de la création»** (art. L.111-2 du CPI, le code de la propriété intellectuelle). Dès que le logiciel commence à être développé, dès qu'il prend forme entre les mains des développeurs, il est protégé, c'est-à-dire que l'auteur peut revendiquer ses droits. **En conséquence, il est essentiel de pouvoir dater cette création**. Une telle preuve peut être faite par tout moyen, même si certains mécanismes dédiés existent. Le dépôt à l'Agence pour la protection des programmes (**APP**) est l'un des plus répandus, mais il est aussi possible de le déposer **devant notaire** ou auprès d'autres organismes spécialisés, voire sur une **forge publique**, tant que l'on peut ultérieurement prouver devant un juge la date certaine du dépôt et l'identité du déposant.

Ce mécanisme de protection immédiate, qui ne nécessite la validation d'aucune autorité extérieure, différencie profondément le droit d'auteur du droit des marques ou des brevets.

Logiciel et brevet

La question de la brevetabilité du logiciel fait l'objet de nombreuses incompréhensions.

Par principe, **le logiciel n'est pas brevetable «en tant que tel»** (art. L. 611-10 CPI). Il n'est pas possible de déposer une demande de brevet qui ne porte que sur un logiciel, pris isolément sans autre élément. Cette interdiction, énoncée tant par la loi que par les juridictions, est fort heureuse et même les États-Unis, traditionnellement en faveur du brevet logiciel, reviennent peu à peu sur leur position. Admettre un brevet ne portant que sur du logiciel reviendrait à admettre un brevet portant sur une fonctionnalité algorithmique. Or, le principe du brevet est de revendiquer une «façon de faire» et pas un «ce que l'on veut faire».

En revanche, le fait qu'un **processus industriel innovant** soit piloté par ordinateur n'est pas en soi un obstacle au dépôt d'un brevet sur ledit processus, dans la mesure où les autres éléments de l'invention obéissent aux conditions de la brevetabilité : **nouveauté, activité inventive, applicabilité industrielle**. L'exemple type est l'invention dite de «contrôle-commande», qui associe la mise en œuvre d'une innovation dans le domaine physique (freinage sans blocage des roues, usinage de précision) à une partie informatique (règles abstraites permettant de contrôler un régulateur sur la base de valeurs fournies par un capteur).

Les parties d'un logiciel protégées séparément ou par d'autres droits

Certains éléments composant un logiciel ne sont pas couverts par le droit d'auteur adapté au logiciel. Attention cependant : certains d'entre eux peuvent être protégés par d'autres outils juridiques.

- Les idées et principes à la base du logiciel ou des fonctionnalités : de façon très classique, **les idées et principes abstraits sont dits «de libre parcours»**, c'est-à-dire que nul ne peut les revendiquer. Ce principe est fondamental. On inclut dans cette catégorie les **algorithmes mathématiques**.
- La **documentation** du logiciel : elle est pour sa part soumise au droit d'auteur classique, celui qui protège les œuvres littéraires et artistiques, mais est parfois aussi assimilée à un «matériel de conception préparatoire».
- Les **œuvres non logicielles** : elles comprennent tous les éléments qui ne relèvent pas du logiciel proprement dit mais qui y sont intégrés, tels que les polices de caractères, les images, icônes et cartes (sous leur forme graphique), les sons et musiques, etc., qui sont protégés par le droit d'auteur classique.
- Les **interfaces homme-machine** (IHM) : l'agencement des éléments de l'interface utilisateur grâce à laquelle l'utilisateur appelle les fonctionnalités du logiciel relève du droit d'auteur générique, en tant qu'œuvre graphique. On dissocie donc la protection du code du logiciel de celle de son IHM. Cette solution est cohérente : dans un livre, les droits de l'auteur du texte et ceux de l'auteur des illustrations sont eux aussi distincts. Les principes d'une IHM, eux, constituent des idées de libre parcours.

- Les **langages de programmation** : comme les langages naturels (qui sont un moyen d'expression mais pas l'expression elle-même) les langages de programmation ne sont pas protégés par le droit d'auteur. Contrairement aux autres solutions juridiques, celle-ci est sans doute la plus contre-intuitive, puisque la complexité ou la richesse nécessaires à l'élaboration d'un langage (et de la « culture » qui l'accompagne) laissent penser que son créateur pourrait prétendre à un droit spécifique.

Autres mécanismes de protection du logiciel

Les autres mécanismes juridiques grâce auxquels les éléments d'un logiciel peuvent être protégés sont très nombreux, et cette protection peut présenter de multiples formes : le **droit des marques**, le recours au secret et au savoir-faire, les actions judiciaires civiles et pénales pour réprimer un acte de concurrence déloyale ou de parasitisme économique qui porterait sur un logiciel, etc. Les bases de données, quant à elles, font l'objet en Europe d'une protection spécifique.

L'auteur

L'auteur : en principe

De prime abord, la règle semble simple : la personne qui produit – de façon originale – un logiciel en est l'auteur et donc le titulaire initial des droits. Elle bénéficie de l'ensemble des droits attachés à cette qualité. Or, la pratique actuelle de conception logicielle est telle que cette règle est très souvent mise à mal. **Il est en effet rare aujourd'hui qu'un logiciel soit conçu, développé,**

analysé, maintenu, mis à jour et amélioré par une personne seule. Dans le cas d'une pluralité d'auteurs, il est souvent possible d'identifier chaque personne chargée d'un bloc spécifique de développements. Chaque contributeur est alors auteur de sa partie, et les droits sur l'œuvre globale s'exercent d'un commun accord entre tous les contributeurs y ayant collaboré.

L'auteur : en pratique

Alors que le droit d'auteur classique considère l'auteur comme un artiste vivant des rentes de sa création, le droit d'auteur adapté au logiciel prend acte de l'organisation industrielle de ce secteur. **Les droits patrimoniaux sur les logiciels créés par des auteurs de logiciels salariés ou agents publics sont automatiquement transférés à leur employeur** (art. L.113-9 CPI). L'employeur, sans être l'auteur du logiciel, en est l'*ayant droit*, c'est-à-dire la personne exerçant les droits patrimoniaux (économiques) sur l'œuvre logicielle, tels que le choix des modes de diffusion, le choix de la licence, etc. Les employés auteurs de logiciels vivent donc de leur travail, mais pas de leurs rentes.

Attention : ceci ne concerne que les salariés et les agents publics, à l'exclusion de tout autre cas, tel que les stagiaires ou les sous-traitants.

Les droits de l'auteur

Les droits d'auteur se répartissent en deux catégories. D'une part, les droits moraux, qui sont une sorte de reconnaissance

juridique du lien « affectif » entre un auteur et son œuvre. D'autre part, les droits patrimoniaux, qui sont la reconnaissance du lien « économique » entre l'auteur et son œuvre.

Les droits moraux

Concernant les logiciels, les droits moraux sont restreints à un droit au nom et un droit à la réputation (art. L. 121-7 CPI). **Tout auteur d'un logiciel dispose du droit absolu à ce que son nom soit mentionné dans le logiciel.** Les « crédits », la documentation, les en-têtes de fichiers sources… Tous les moyens sont bons ! Quant au **droit à la réputation**, celui-ci permet en théorie d'empêcher la modification de son logiciel si celle-ci porte atteinte à la réputation de son auteur.

Les droits patrimoniaux

Les droits patrimoniaux rassemblent toutes les prérogatives de l'ayant droit d'un logiciel de procéder à l'exploitation de son logiciel : **exploitation, diffusion sous certaines licences, cession de licences, cession de ses droits**. Par exemple, le choix de diffuser son logiciel en open source, qui est une forme incontestable d'exploitation, est un exercice de ses droits patrimoniaux.

Comme expliqué plus haut, l'auteur peut ne pas être l'ayant droit du logiciel qu'il a écrit, et vice versa, dès le moment où les droits nés sur la tête de l'auteur ont été transférés à autrui.

Chapitre 2

Cadre juridique spécifique du logiciel libre

Contrairement à une idée reçue, le logiciel libre n'est pas caractérisé par l'absence de droits mais plutôt par une nouvelle façon d'utiliser ses droits. Le cadre juridique habituel de diffusion du logiciel est orienté du point de vue des ayants droit. Le mouvement des logiciels libres inverse cette logique, en abordant ce cadre juridique **du point de vue de l'utilisateur**.

Le caractère libre d'un logiciel est déterminé par la licence sous laquelle il est diffusé : celle-ci est qualifiée de **libre** si elle garantit aux utilisateurs les libertés définies par la Free Software Foundation (FSF), et de **open source** si elle répond aux dix critères de l'*Open Source Definition*, établis par l'Open Source Initiative (OSI).

Logiciel libre (FSF) & open source (OSI)

Sous l'impulsion de Richard M. Stallman[1], la **Free Software Foundation** a établi dès 1985 une définition du logiciel libre qui a convergé autour de quatre libertés garanties à l'utilisateur :

1. On lira avec profit *Richard Stallman et la révolution du logiciel libre – Une biographie autorisée*, R. Stallman, S. Williams, C. Masutti, Eyrolles, 2013.

- la liberté d'**exécuter le programme, sans restriction** et pour n'importe quel usage (liberté 0) ;
- la liberté d'**étudier** le fonctionnement du programme et de le **modifier** pour qu'il effectue des tâches informatiques comme souhaité (liberté 1) ; l'accès au code source en est une condition nécessaire ;
- la liberté de **redistribuer** des copies, donc d'aider son prochain (liberté 2) ;
- la liberté de **distribuer aux autres des copies des versions modifiées** (liberté 3) ; ce faisant, on donne à toute la communauté la possibilité de profiter des changements apportés ; l'accès au code source en est une condition nécessaire.

En 1998, Bruce Perens et Eric S. Raymond fondèrent l'**Open Source Initiative** (OSI) pour faciliter l'adoption du logiciel libre dans un plus grand nombre de contextes, en lui appliquant un *rebranding* pour lever l'ambiguïté du terme *free* en anglais, qui peut signifier à la fois « gratuit » et « libre », et pour mettre l'accent sur les aspects techniques afin de s'adapter à des contextes où la portée militante pouvait constituer un frein à l'adoption des solutions. L'OSI a alors proposé une définition de l'open source basée sur dix critères, déclinant les quatre libertés formulées par la FSF. Cette approche se traduit dans l'**Open Source Definition** :

1. Liberté de redistribution (gratuite, commerciale, etc.) ;
2. Fourniture du code source ;
3. Possibilité de créer des œuvres dérivées et de les diffuser sous des conditions identiques ;
4. Respect de l'intégrité du code source de l'auteur – la licence peut restreindre la redistribution d'œuvres dérivées à condition

d'autoriser la distribution de fichiers « rustines » (*patches*) applicables au programme lors de son installation ;

5. Pas de discrimination à l'encontre de personnes ou de groupes de personnes ;

6. Pas de discrimination contre un champ d'application particulier ;

7. Critère de la distribution – la licence est appliquable dès lors qu'il y a distribution (afin d'éviter des restrictions additionnelles non prévues par la licence) ;

8. La licence ne doit pas être spécifique à un produit donné ;

9. La licence ne doit pas imposer de restrictions sur les logiciels distribués avec le logiciel licencié ;

10. Neutralité technologique – la licence ne doit dépendre d'aucune technologie particulière.

Il existe bien quelques licences qui correspondent à la définition de l'OSI sans répondre à celle de la FSF, mais elles restent marginales. Dans la pratique, lorsqu'il y a une opposition entre les termes, elle est davantage utilisée pour marquer une différence d'approche philosophique qu'une distinction juridique. Les deux concepts sont considérés comme équivalents et sont souvent regroupés au sein du terme **FOSS** (*Free and Open Source Software*) ou **FLOSS** (*Free/Libre and Open Source Software*).

Standardisation des licences : une force juridique

Les textes juridiques implémentant ces principes, c'est-à-dire les licences libres, présentent la particularité d'être le plus souvent

indépendants d'un logiciel particulier, à la différence des licences privatives propres à chaque éditeur. **Cette standardisation a un effet bénéfique sur la pratique de la conformité juridique**, parce qu'elle rend la connaissance des conditions d'exploitation d'un logiciel bien plus simple pour l'utilisateur ou le contributeur. **De plus, ces textes, pour les plus courants d'entre eux, ont été validés par différents tribunaux, et offrent donc une grande sécurité juridique**.

Bien qu'on ait observé une **multiplication des licences libres** (le projet SPDX en recense plus de 300 !) venant diluer ces avantages, la majorité des projets libres sont couverts par un ensemble réduit de celles-ci, les principales étant les suivantes : GNU General Public Licence (**GPL**), Lesser GNU General Public Licence (**LGPL**), GNU Affero General Public Licence (**AGPL**-3.0), **MIT**, **Apache License** 2.0, **BSD**-3-Clause, **Mozilla Public Licence** (MPL), **Eclipse Public License** 1.0.[2]

Certaines licences ont été rédigées pour répondre à des problématiques spécifiques. C'est par exemple le cas des licences **CeCILL**, rédigées afin que leur version française puisse explicitement faire foi auprès des tribunaux français, même si ceux-ci ont aussi pu reconnaître des licences rédigées en anglais telle que la GNU GPL.

---

2. « Le lecteur curieux pourra consulter ce message de juin 2010 qui donnait la liste des licences utilisées par les 29 000 logiciels empaquetés par Debian : https://lists.debian.org/debian-policy/2010/06/msg00099.html
En synthétisant et après avoir écarté les paquets de documentation et de données placés sous les licences libres spécifiques (GFDL, CC-By, CC-By-SA), on constate qu'environ 60 % des logiciels libres sont diffusés sous l'une des versions de la licence GNU GPL et 20 % sous l'une des versions de la licence GNU LGPL. Sébastien Dinot avait calculé à l'époque qu'avec 10 licences, on couvrait en effet plus de 98 % des cas ! »

Dans l'écosystème des licences libres, une licence peu usitée risque d'autant moins d'être choisie qu'elle contient des clauses incompatibles avec les licences dominantes. C'est pour cela que la majorité de ces licences contiennent des **clauses de compatibilité explicite** avec les licences les plus courantes, résolvant ces questions dès l'origine.

Des dispositions permettent également de «mettre à jour» automatiquement le texte de la licence d'un logiciel vers une version ultérieure de celle-ci, sans que l'ayant droit ait besoin d'en redistribuer le code source. Tel est le sens de la mention «GPLv2 + » qui permet à la personne recevant le logiciel de considérer qu'il est couvert par la GPL v2 **ou toute autre version ultérieure**. Ceci est d'autant plus important que les licences GPL v2 et v3 sont incompatibles entre elles.

Droits et obligations

Types de droits accordés

Les licences logicielles libres se situent par nature sur le plan du droit d'auteur, et c'est à ce niveau que sont accordées les libertés fondamentales qui les caractérisent. Cependant, les **brevets logiciels** constituant une réalité incontournable de certaines juridictions, dont les États-Unis, certaines licences accordent également des droits sur ce plan. La distinction est alors explicitement présente dans la licence.

Extrait de la licence Mozilla Public License 2.0 :

```
2. License Grants and Conditions
   2.1. Grants
   Each Contributor hereby grants You a world-wide,
royalty-free, non-exclusive license:
    (a) under intellectual property rights (other than
patent or trademark) Licensable by such Contributor
to use, reproduce, make available, modify, display,
perform, distribute, and otherwise exploit its
Contributions, either on an unmodified basis, with
Modifications, or as part of a Larger Work; and
   (b) under Patent Claims of such Contributor to
make, use, sell, offer for sale, have made, import,
and otherwise transfer either its Contributions or its
Contributor Version.
```

Obligations passives et actives

À l'exception des licences s'apparentant à l'expression d'une contribution volontaire au domaine public, les droits accordés par les licences libres sont soumis à un certain nombre d'obligations, compatibles avec les libertés fondamentales qu'elles accordent. Ces obligations peuvent être **actives**, quand elles indiquent une action à accomplir afin d'être en conformité avec la licence, ou **passives**, quand elles demandent de ne pas effectuer certaines actions pour demeurer en conformité avec la licence.

Il est donc essentiel pour toute entité réutilisant du code open source de mettre en œuvre les processus lui permettant de connaître les obligations auxquelles le code qu'elle produit ou redistribue est soumis et de s'en acquitter selon les modalités requises.

Parmi les exemples courants d'obligations passives, on trouve **celle de ne pas utiliser le nom de l'auteur pour la promotion**

d'un produit tiers ou de ne pas attaquer les utilisateurs du logiciel sur la base de brevets.

Extrait de la licence BSD 3-clause « New » ou « Revised »

```
 3.Neither the name of the copyright holder nor the
names of its contributors may be used to endorse or
promote products derived from this software without
specific prior written permission.
```

Extrait de la licence Apache 2.0

```
If You institute patent litigation against any entity
(including a cross-claim or counterclaim in a lawsuit)
alleging that the Work or a Contribution incorporated
within the Work constitutes direct or contributory pa-
tent infringement, then any patent licenses granted to
You under this License for that Work shall terminate
as of the date such litigation is filed.
```

Les obligations actives les plus courantes concernent la **reconnaissance de paternité** de l'œuvre selon des formalismes qui peuvent varier (dans le logiciel lui-même, dans la documentation, etc.).

Extrait de la licence MIT

```
   The above copyright notice and this permission
notice shall be included in all copies or substantial
portions of the Software.
```

Les obligations actives les plus structurantes sont celles liées au « copyleft ».

Obligation de réciprocité : le copyleft

Le **copyleft** (terme né d'un jeu de mots sur le terme *copyright*, et parfois traduit par «gauche d'auteur») désigne **l'obligation de diffuser les œuvres modifiées sous la même licence que l'œuvre initiale**. On parle d'obligation de réciprocité, qui garantit les mêmes droits à toute la chaîne des utilisateurs et contributeurs successifs. Ces dispositions favorisent la convergence des efforts de développement autour d'un projet commun, en empêchant la création de versions modifiées privatives concurrentes de la version partagée par tous.

Les licences copyleft ont prouvé leur efficacité à fédérer des acteurs divers et même potentiellement concurrents, comme l'atteste le succès de projets industriels de très large envergure, tel le noyau Linux pour ne citer que lui.

La licence emblématique du copyleft est la GNU General Public Licence (GPL), mais d'autres licences comportent ce type de clause : l'Eclipse Public License (EPL), la European Union Public License (EUPL), etc.

Le copyleft est un critère majeur de classification des licences, car il conditionne les modèles d'affaires associés et les possibilités de réutilisation dans des projets tiers.

Degrés de copyleft

De façon schématique, il existe trois degrés de copyleft selon les conditions applicables à la **redistribution des modifications** du logiciel couvert et/ou de son **intégration** dans des

projets plus importants **par combinaison** avec des modules tiers couverts par d'autres licences.

Copyleft fort (GNU GPL, CeCILL-A[3] 2.1, etc.)

Ces licences, parfois dites « diffusives », **imposent leurs termes à toute œuvre dérivée** issue du logiciel. Pour autant, cela ne signifie pas que tous les modules tiers composant le logiciel dérivé doivent être rendus disponibles sous cette même licence ; il est suffisant que les licences de ces modules soient compatibles avec la licence copyleft concernée. En particulier, **le respect des termes des licences à copyleft fort impose que les licences des autres modules soient libres**, afin de garantir les libertés des usagers sur l'ensemble du logiciel. L'obligation d'utiliser exactement la même licence ne s'applique qu'aux seules modifications internes au logiciel couvert.

Copyleft faible (GNU LGPL, CDDL, Mozilla MPL, CeCILL-C, etc.)

Ces licences permettent de **combiner le code qu'elles couvrent avec des modules tiers placés sous tous types de licences, y compris privatives**, à condition qu'il n'existe pas d'incompatibilité entre les licences. Cela permet de s'assurer que les évolutions du logiciel initialement couvert resteront toujours sous licence libre. Certaines licences à copyleft faible imposent en outre que le logiciel dérivé puisse être reconstruit à partir d'une nouvelle version du logiciel libre, afin que les améliorations apportées au logiciel couvert puissent bénéficier au logiciel combiné.

3. La dénomination « CeCILL-A » est souvent utilisée pour désigner la licence CeCILL en la distinguant des licences CeCILL-B et CeCILL-C. http://cecill.info/

Non-copyleft (MIT, Apache, BSD, CeCILL-B, etc.)

Ces licences, dites « permissives », n'ont pas d'effet copyleft. Elles permettent de modifier le code ou de l'intégrer dans un logiciel distribué sous tout autre type de licence, sans que l'usager puisse obtenir le code source de ce logiciel et exercer les droits initialement attachés à la licence libre. Ce type de licence est à l'origine du fait que **l'immense majorité des logiciels actuels – libres ou non – repose en partie sur du code initialement libre**.

Éléments déclencheurs : diffusion ou distribution par le réseau

L'exécution des obligations d'une licence est associée à différents **éléments déclencheurs,** qui sont le plus fréquemment liés à la **distribution du logiciel**. Ce point est particulièrement sensible dans le contexte des **services Web (SaaS)**, où l'utilisateur n'est pas tenu de télécharger le logiciel pour en utiliser les fonctions, puisqu'elles lui sont fournies via le réseau. L'absence de redistribution des logiciels utilisés pour produire ces services permettait de contourner les clauses de réciprocité des licences copyleft classiques telles que la GNU GPL. En réaction à cela, des licences ont été rédigées pour **définir l'accès par le réseau comme élément déclencheur de la licence**, afin de donner à l'utilisateur du service les mêmes droits qu'à l'utilisateur direct du logiciel, et notamment l'accès au code source. L'exemple canonique de ce type de fonctionnement est l'**Affero General Public License (AGPL)**.

Non-respect des licences libres

Tout comme les licences privatives (propriétaires), les licences libres sont des outils de gestion de droits : **la violation de leurs clauses peut entraîner des sanctions parfois lourdes.** Si l'on constate dans les communautés logicielles libres une préférence pour la remédiation au contentieux (ce que reflètent les licences de la FSF, qui incluent des délais de remise en conformité), le recours aux tribunaux n'est pas inhabituel. De la **condamnation pour contrefaçon ou concurrence déloyale** à l'**indemnisation des ayants droit** du logiciel, en passant par l'**obligation de communiquer ses propres codes sources**, les sanctions peuvent être lourdes.

Marques, logos et logiciels libres

Les licences libres ne comprennent souvent volontairement pas de disposition concernant les marques. C'est cependant une dimension clé dans le développement des projets libres et le sujet est souvent traité dans un document spécifique distinct.[4]

Ces dispositions annexes concernant **l'usage du nom et des logos** permettent de protéger les utilisateurs finaux contre la diffusion de versions dérivées contenant des éléments nuisibles (adware, malware, etc.) et participe de la réputation des éditeurs de logiciels FLOSS.

4. Le Comité de pilotage du Groupe thématique Logiciel libre conseille d'ailleurs aux partenaires de projets de R&D collaborative en logiciel libre de prévoir dès l'accord de consortium des dispositions concernant l'exploitation du nom du projet.

Cette **décorrélation entre droit d'auteur et marques** peut également être le fondement de **modèles économiques viables** : c'est par exemple le cas de Red Hat, qui propose une distribution commerciale Red Hat Enterprise Linux, qui cohabite avec une version compatible au niveau binaire, mais dépourvue des éléments de marque : CentOS.

Chapitre 3

Usage et exploitation

Droits et obligations en cas d'intégration de logiciel libre

Il est nécessaire de connaître ses droits et obligations lorsqu'on intègre du logiciel libre dans son propre logiciel, et que l'on crée ce que nous appellerons un «logiciel dérivé». Cela nécessite au préalable de connaître le mode d'utilisation ou d'exploitation de ce logiciel dérivé. Nous en distinguerons plusieurs types.

Usage interne à l'entreprise

Le contexte est celui du système d'information propre à l'entreprise, des outils de développement ou de test, ou des logiciels de production. Cela concerne également les logiciels qui sous-tendent des offres commerciales de service outillé. **Ce type d'usage au sein d'une unique personne morale n'impose aucune contrainte**, quelle que soit la ou les licences des composants logiciels intégrés. En particulier, il n'entraîne **pas d'obligation de redistribution du logiciel dérivé**. On voit ici toute la puissance du logiciel libre pour ces cas d'usage. Cependant, **la distribution du logiciel à des entreprises sous-traitantes et/ou des filiales constitue dans la majorité des cas une redistribution à des tiers.**

Offre en mode SaaS

Le SaaS (*Software as a Service*), autre mode d'exploitation d'un logiciel dérivé, mérite qu'on s'y attarde. Le logiciel n'est certes pas distribué, puisqu'il s'exécute sur les serveurs de l'entreprise éditrice du logiciel dérivé. En conséquence, **pour la plupart des licences libres, le SaaS est assimilé à un usage interne**, sans aucune contrainte. Les licences Affero-GPL (**AGPL**), **EUPL**, ou encore Apple Public Source License (**APSL**), etc., considèrent en revanche que cet usage est une forme d'interaction avec le logiciel et **imposent que le logiciel dérivé soit distribué aux usagers**. Notons cependant qu'il n'est pas impossible que, pour fonctionner, le logiciel SaaS nécessite l'exécution sur le poste de l'utilisateur de certains codes, tels que des scripts «côté client». Dans ce cas, ces codes sont bien distribués, ce qui peut déclencher les clauses de la licence libre (notamment le copyleft), parfois même sur l'ensemble du logiciel.

Vente de licences sous forme de licence privative (communément appelée «licence propriétaire»)

Cela concerne principalement les entreprises qui vendent des progiciels (contraction des termes «produit» et «logiciel») sous licence privative. Seul un exécutable est livré aux clients de ces entreprises. Dans ce cas, **il est considéré qu'il y a distribution du logiciel dérivé**. Les licences des composants logiciels intégrés dans le logiciel dérivé peuvent avoir un effet.

Comme nous l'avons vu, la capacité à incorporer des modules sous licences libres au sein d'un logiciel que l'on veut distri-

buer sous licence privative dépend du type de copyleft de la licence. Le choix des modules peut donc conditionner fortement le type de modèle économique de l'entreprise éditrice.

Vente de produits matériels dans lequel du logiciel est embarqué

Cela couvre les robots, les capteurs, les objets connectés (IoT), etc. Il est avéré que **la vente de produits est une distribution**, et les droits et obligations sont les mêmes que dans le cas de vente de licence sous forme de licence privative. **La location de produits doit également être considérée comme une distribution**, bien que cela puisse encore être discuté dans certains cas.

Diffusion sous licence libre

La licence des composants libres qui vont être intégrés dans le logiciel dérivé influe sur les droits applicables à la distribution de ce logiciel. Prenons par exemple un logiciel dérivé auquel on souhaite faire bénéficier des termes d'une licence permissive, de type BSD, Apache, etc., afin qu'un tiers puisse «refermer» le logiciel. Ce logiciel dérivé ne pourra intégrer de composants sous licence copyleft, de type GPL, puisque le composant sous licence copyleft imposerait que le logiciel dérivé soit toujours distribué sous des termes permettant l'accès au code source[5].

5. Le livre *Option libre : du bon usage des licences libres,* B. Jean, Framasoft, 2011 présente une synthèse des compatibilités entre principales licences libres. On pourra également se référer à l'ouvrage de référence *Droit des logiciels*, F. Pellegrini, S. Canevet, PUF, 2013.

Choix de la licence d'exploitation

Nous abordons ici les choix de la licence en fonction du modèle économique d'exploitation. Par exploitation, on entend exploitation industrielle ou commerciale par une entreprise et également exploitation des résultats par un organisme de recherche. Cette exploitation est également appelée ***valorisation***.

Le choix de licence passe par deux étapes :

- Choisir le(s) modèle(s) d'exploitation (sous licence privative et/ou sous licence libre), éventuellement simultanément sous plusieurs licences ;
- Si une distribution sous licence libre est choisie, sélectionner licence libre adaptée.

Le processus de choix de licence passe par les étapes suivantes : approche itérative entre la stratégie d'exploitation et le périmètre du logiciel, faisabilité, corrections éventuelles, packaging et diffusion, mise en place de la gouvernance projet.

Ces réflexions doivent être menées le plus en amont possible du développement logiciel, car elles peuvent conditionner le choix des composants logiciels et les coûts de (re)développement. Cette approche itérative n'a en général pas lieu au sein des entreprises, où le choix d'une licence est effectué dès le départ. On l'observe le plus souvent dans les logiciels issus de travaux de recherche qui seront diffusés sous licence libre mais dont le développement a été effectué de manière incrémentale et sans prendre en compte ces considérations.

Approche itérative entre la stratégie d'exploitation et le périmètre du logiciel

Pour les académiques, nous préférerons remplacer la notion de stratégie d'exploitation par celle d'**intention**. En effet, il faut avoir à l'esprit qu'on a constaté en moyenne que 10 années étaient nécessaires entre la première diffusion d'un logiciel et sa réelle exploitation industrielle ou commerciale par des entreprises.

On travaillera alors à définir ce qu'on attend d'une diffusion sous licence libre :

- Obtenir des retours de la communauté ?
- Faciliter le test d'un nouveau logiciel et accélérer l'expérimentation à faible coût ?
- Rechercher une standardisation logicielle (interopérabilité) ?
- Partager les coûts et les risques (mutualisation) ?
- etc.

Une approche itérative est nécessaire entre la stratégie d'exploitation et l'objet qui sera diffusé :

- Quel est le marché visé ?
- Le logiciel a-t-il une valeur métier très forte ?
- Ou a-t-il une grande généricité avec une forte potentialité de communauté ? etc.

Le périmètre du logiciel diffusé devra être bien circonscrit afin de permettre la définition d'une stratégie fine de licence : telle partie du logiciel sera sous une certaine licence, telle autre

sous une autre licence. Encore une fois, nous rappelons que peuvent cohabiter différents modèles de licences sur un même logiciel : libre/libre, libre/privatif.

Si l'on opte pour une licence de type libre, reste à choisir laquelle. Nous proposons ici quelques grandes lignes à adapter au cas par cas :

- **Si le logiciel est un logiciel applicatif**, une licence de type GPL/CeCILL-A ou encore Affero GPL, est préconisée. Elle favorise au maximum les retours de la communauté des utilisateurs et des contributeurs du logiciel. Dans le cadre de logiciels issus de projets de recherche pour lesquels l'exploitation aura lieu plus tard, cette catégorie de licence demeure en outre compatible avec une future valorisation au moyen de licences privatives, par le biais d'un schéma de double licence. Concrètement, ce mécanisme de double licence n'est possible que si l'éditeur académique, puis éventuellement commercial, demande tout au long du développement à tous les contributeurs un transfert de leurs droits patrimoniaux.
- **Si le logiciel est une bibliothèque ou un composant**, une licence de type LGPL ou CeCILL-C est préconisée. Elle permet d'une part l'accès aux modifications par des tiers et d'autre part une liberté d'utilisation par la majorité des acteurs économiques.
- **Si le logiciel est une commodité scientifique**, ou encore un bien commun, une licence de la famille BSD ou CeCILL-B est préconisée, pour une diffusion maximale.

Bien entendu, il faut également tenir compte de la culture de la communauté cible. Par exemple, les logiciels d'infrastructure de type « cloud » diffusés sous licence libre le sont très souvent sous licence permissive de type Apache ou MIT.

Faisabilité

Autant que possible, les questions juridiques doivent être posées voire résolues **au démarrage du processus industriel**. En effet, il est essentiel de mener ce type de réflexion concurremment au développement du projet, puisqu'elles le conditionnent partiellement.

Dans le cas de logiciel issu de la **recherche publique**, il arrive que le choix de la licence se pose tardivement, lorsque le logiciel atteint une certaine maturité. Il est alors nécessaire de réfléchir au choix de la licence en faisant autant que possible abstraction de la licence des composants logiciels intégrés. Par exemple, le fait d'utiliser un module sous GPL n'impose pas de diffuser les autres modules sous la même licence, mais seulement sous des licences libres compatibles, quelles qu'elles soient.

Nous proposons donc de commencer par choisir une licence puis, une fois ce choix fait, libre ou non, de vérifier s'il est possible de le mettre en pratique.

Dans le cas contraire, il faudra parfois régler des questions d'ayants droit ou encore de compatibilité de licence des composants préexistants avec la licence de diffusion privative ou libre souhaitée.

Il existe de nombreuses possibilités de correction :

- réécriture,
- acquisition des droits,
- demande aux ayants droit d'un changement de licence,
- renoncement à la diffusion d'un composant incompatible,
- signature d'un accord d'indivision,
- etc.

Ces méthodes correctives ne sont pas idéales, car elles induisent un fort surcoût.

Et la suite...

Il sera nécessaire de **mettre en place la gouvernance du patrimoine immatériel** (licences des composants utilisés, centralisation des droits) **et d'une éventuelle protection de la marque et du logo**.

Conclusion

La maîtrise des aspects juridiques du logiciel libre est essentielle pour bénéficier du colossal patrimoine technologique et d'innovation disponible dans le monde, et ainsi accélérer le développement et la qualité de ses propres logiciels.

Cette maîtrise est également indispensable pour accompagner dès leur démarrage les projets collaboratifs visant à construire une stratégie d'exploitation sous forme de licence libre, et tirer parti de tous les bénéfices du logiciel libre : modèle économique innovant, accélération de la mise sur le marché et communauté de développement.

À PROPOS DU GROUPE THÉMATIQUE LOGICIEL LIBRE DE SYSTEMATIC (GTLL)

Créé en 2007, le Groupe thématique Logiciel Libre du pôle Systematic Paris-Region forme l'un des principaux viviers de l'Open Source en France.

Avec pour mission de **développer l'écosystème du libre en Île-de-France**, le GTLL regroupe plus d'une centaine d'acteurs de l'innovation ouverte (PME, ETI, grands groupes et académiques). Il vise à favoriser la coopération, l'innovation et l'emploi, autour de projets de R&D collaborative et grâce à des actions de soutien au développement des entreprises innovantes (promotion, marketing, stratégie, aide à la recherche de financements...), dans le cadre des principes et des valeurs de l'open source. Il est à ce jour le plus important cluster au monde à focaliser ses activités de R&D collaborative sur les logiciels libres et les défis spécifiques à l'open source, comme l'after PC (l'ère informatique du Cloud, des mobiles et des objets connectés), la qualité logicielle, et le déluge des données. *Après 9 ans d'existence, 57 projets de R&D collaborative consacrés au logiciel libre représentant un effort de R&D de près de 199,9 M€ ont déjà été financés grâce à l'aide du GTLL.*

L'action du Pôle est soutenue par :

Partenaires stratégiques : Deloitte. Systemx GROUPE FG DESIGN

Par les propriétés juridiques qu'elles instaurent, les licences libres et open source permettent de clarifier et d'industrialiser les modes de répartition de valeur et de collaboration autour du logiciel.

Réfléchir sur le logiciel libre, c'est réfléchir en termes de valeur et de valorisation, et définir les droits et obligations qui encadrent la répartition de cette valeur.

Armés des principes exposés dans cet ouvrage, entrepreneurs, développeurs et architectes comme chercheurs, sauront maîtriser la complexité inhérente à l'assemblage de centaines de composants open source dans le cadre de projets d'envergure. Les éditeurs de logiciel libre trouveront également une aide aux choix de licence, en fonction des modes de collaboration, de partage de valeur et de gouvernance qu'ils souhaitent privilégier.

Ce guide publié par le GTLL de Systematic en collaboration avec le cluster Aquinetic vise à éclairer les mécanismes juridiques liés à la valorisation des logiciels libres et open source – et, partant, de tous les logiciels.

Le **Groupe thématique Logiciel Libre** forme l'un des principaux viviers de l'open source en France. Il rassemble startups, PME, grands groupes, universités et centres de recherche autour d'une même vision des défis technologiques de demain et d'un engagement profond pour la compétitivité de notre économie.

Ouvrage réalisé en collaboration avec **Aquinetic**, cluster aquitain partenaire spécialisé sur l'open source.

Aquinetic
INNOVATION LOGICIELS LIBRES EN AQUITAINE

Systematic
Paris Region Digital Ecosystem

Conception : Nord Compo