http://www.droit-technologie.org

Présente :

La licence publique générale : un système original de protection juridique pour les créations issues des systèmes de développement coopératifs

Jean-Baptiste Soufron

Sous la direction de Monsieur le Professeur Théo Hassler
DESS Multimédia et Systèmes d'Information
Université Strasbourg III Robert-Schuman

jean-baptiste.soufron@libertysurf.fr

Date de mise en ligne : juin 2002

La Licence Publique Générale : un système original de protection juridique pour les créations issues des systèmes de développement coopératifs.

Jean-Baptiste Soufron
Sous la direction de Monsieur le Professeur Théo Hassler

DESS Multimédia et Systèmes d'Information
Université Strasbourg III Robert-Schuman

ce document est disponible en html à l'adresse suivante :
`soufron.free.fr/files/gpl.html`

4 février 2002

INTRODUCTION

L'amélioration moderne des réseaux de communication a eu comme corollaire une diminution du coût des communications alors même que leur rapidité et leur fiabilité s'amélioraient régulièrement. Aujourd'hui, des individus géographiquement dispersés travaillant sur des créations communes sont capables de collaborer virtuellement sans avoir besoin d'une structure hiérarchique. Intuitivement, on comprend bien que plus on peut augmenter l'interaction entre ces collaborateurs, plus ils partagent leurs connaissances et plus leur capacité de création sera importante, autant en termes qualitatifs que quantitatifs. Désormais, la création passe par la communication et le partage autour d'un projet commun plus que par l'introspection et la recherche en solitaire : la nouvelle règle est celle des systèmes de développement coopératif en réseaux ou *"virtual decentralized network"* [Dafermos, 2001].

Les premiers à bénéficier de ce nouveau paradigme ont bien sur été ceux qui furent à l'origine de la mise en place des nouveaux moyens de communication. Les programmeurs informatiques ont commencé très tôt à expérimenter cette nouvelle façon de travailler au fur et à mesure qu'ils créaient les outils pour permettre aux autres de le faire. En partageant toutes leurs découvertes l'innovation et la création ont été surmultipliées : malgré les difficultés géographiques et en dépit des principes hiérarchiques traditionnels, l'informatique libre n'a jamais cessé de croître depuis 25 ans sans aucun système d'organisation central, jusqu'à créer Internet, la plus grande source d'innovation et de Création de notre époque.

Mais les chercheurs ont rapidement été confrontés à un grave problème : la loi ne les considérait pas comme des auteurs et ne leur permettait pas de pérenniser leurs travaux. Face à la ré-appropriation de leurs recherches par les propriétaires légaux de leurs oeuvres ils ont du imaginer de nouvelles façons de protéger leurs logiciels en organisant l'apport de tous les participants autour d'un projet et en garantissant que celui-ci irait jusqu'à son terme. La solution la plus simple était de placer leurs relations sous l'égide d'un contrat de licence d'utilisation car la protection classique du droit d'auteur s'était révélée inadaptée. C'est l'origine de la naissance des licences de type Open Source dont le modèle le plus abouti et le plus étudié est la Licence Publique Générale GNU ou GPL qui régit l'utilisation de la majorité des Logiciels Libres.

Il faut donc comprendre comment les systèmes de développement coopératif se sont imposés comme la solution évidente aux problèmes posés par les autres structures de management et pourquoi la GPL est tellement adaptée pour protéger les logiciels qui sont crées selon ces systèmes.

Chapitre 1

Le développement d'une nouvelle forme de management et de création : le développement coopératif

La vraie qualité de l'échange ce n'est pas qu'il implique 2 gains, c'est qu'il offre plus à partager pour tous [Bastiat, 1851, chap. 5 de l'échange].

Les stratégies de management ont beaucoup évoluées au cours du siècle dernier. Pour les entreprises, la révolution industrielle a augmenté la valeur des connaissances et l'amélioration des moyens de communication a offert une plus grande souplesse dans la gestion des projets : c'est l'introduction du développement coopératif ou *"virtual decentralised network"* [Dafermos, 2001]. Les logiciels libres sont un exemple classique de cette forme d'organisation plus naturelle et plus humaine.

1.1 L'évolution des formes de management : d'une structure hiérarchique vers une structure en réseaux.

"Free Markets have taught that innovation is best when ideas flow freely" Adam Smith

1.1.1 La théorie de départ du management : la chaîne hiérarchique

Pendant longtemps le principe de la chaîne hiérarchique a prévalu comme modèle d'organisation des entreprises mais au fur et à mesure que l'idéal de la production de masse s'effaçait, on a vu apparaître de nouvelles formes de management.

1.1.1.1 La découverte de la chaîne hiérarchique

Issu du monde militaire, le principe de l'organisation hiérarchique fonctionne quand 3 hypothèses sont réunies :

- les processus à mettre en oeuvre doivent être connus : il faut une méthode
- l'environnement dans lequel ils prennent place doit être défini dès le départ : il faut un marché
- le résultat final à obtenir doit être prévu à l'origine du projet : il faut un produit

En 1911, F.W. Taylor écrit "The principles of Scientific Management" sur l'organisation scientifique du travail. Son idée principale était que la connaissance détenue par les ouvriers (c'est à dire le savoir-faire, les méthodes de travail, les techniques, etc...) devait être exprimée de façon explicite, mise sur le papier pour pouvoir être contrôlée et remonter au niveau des dirigeants qui pourront alors la redistribuer de façon plus efficace au sein de l'entreprise. Il voit un projet comme une machine dirigée par une chaîne hiérarchique ou les ouvriers n sont que de simples rouages spécialisés et déshumanisés.

Peu de temps après, déjà, H. Fayol, décrivit le principal problème de la chaîne hiérarchique : la remontée et le contrôle de la connaissance au niveau des dirigeants produit un allongement de la chaîne de décision[1]. Il serait beaucoup plus intuitif de communiquer au sein d'un même niveau hiérarchique plutôt que d'être obligé de remonter vers les niveaux supérieurs. Malgré son intuition de ce problème, H. Fayol estimait que l'organisation hiérarchique était inévitable quand un projet commençait à prendre une certaine ampleur et qu'une production à la chaîne était l'objectif recherché [Fayol, 1949, chap. 4].

1.1.1.2 La domination du principe de la structure hiérarchique

Mais Weber, le fondateur moderne du management, imposa l'idée qu'une organisation bureaucratique est la seule organisation rationnelle pour une entreprise et que toutes les autres formes de management ne sont que des illusions.

L'histoire a finalement montré que la structure hiérarchique n'est la meilleure organisation que tant que l'on reste dans le domaine de la production de masse industrielle. Dès 1962, un historien du management appelé A. Chandler a commencé à pointer les carences de cette organisation en étudiant la structure de General Motors. Selon ses conclusions, plus la taille d'une société augmente et plus sa hiérarchie devient incapable de contrôler et de coordiner l'organisation de ses employés. Une solution semblait être la délégation de compétences au sein de la chaîne hiérarchique pour essayer de la raccourcir artificiellement [Chandler, 1962, p382-383].

[1] il suffit d'imaginer un triangle équilatéral dont le sommet serait le dirigeant et dont les deux angles seraient les ouvriers : l'information met plus de temps à aller d'un ouvrier à l'autre en remontant la chaîne hiérarchique vers le dirigeant puis en la descendant vers le second ouvrier qu'en allant directement de l'un à l'autre.

1.1.2 La découverte des structures de développements en réseaux

La croissance du marché n'a pas tardé à montrer les limites des structures de management hiérarchiques et il a fallu commencer à imaginer des formes de management différentes comme les structures de développement en réseaux.

1.1.2.1 La prise de conscience de la nécessité d'une autre forme de management

Jusqu'à cette époque toutes les structures de management cherchaient à obtenir une production de masse où les employés-rouages devaient travailler sur des chaînes d'assemblage, le mot-clé était encore celui de Ford : *"mass production for mass market"*.

Mais, après l'étude de Chandler, il devenait évident que la chaîne hiérarchique n'existait que parce qu'on était persuadé à tort que la seule personne destinée à tout connaître de la compagnie et capable de le faire devait être son dirigeant.
Il existait pourtant deux formes de management imaginables : le système mécanique classique et un système organique. Une organisation mécanique est appropriée à des conditions de production stables et prévisibles alors qu'un système organique est idéal quand les conditions de production sont sujettes à des évolutions imprévisibles. Plus l'environnement d'une entreprise est complexe, plus sa structure devrait être flexible pour réagir rapidement aux changements de marché qui peuvent se présenter.

1.1.2.2 Du *"Total Quality Management"* au principe de la *"Learning Structure"*

L'inertie des mentalités a empêché l'évolution des structures de management là où elles avaient pourtant été disséquées et critiquées.

En fait c'est le japonais Sakichi Toyota qui comprit le premier que la flexibilité et la vitesse de réaction ne pouvaient être obtenus qu'à la condition d'établir des relations étroites et sur un pied d'égalité avec ses fournisseurs, ses sous-traitants et ses employés. Plutôt que de suivre le modèle de la chaîne hiérarchique verticale, Sakichi Toyota décida de pratiquer le système de la production en temps réel pour minimiser le coût de la création et accélérer l'innovation. L'idée principale de Toyota pour atteindre son objectif a été, tout en respectant le principe de la division du travail de Taylor, de déléguer son pouvoir à ses employés : Toyota commença à répartir le travail en plusieurs projets indépendants plutôt qu'en plusieurs postes au sein de la chaîne d'assemblage. De cette façon, un employé pouvait passer d'un projet à l'autre ou réutiliser ce qu'il avait appris quelque part dans un projet différent. Cette innovation du management a permis à Toyota d'améliorer la qualité de ses produits simplement en multipliant l'innovation et la communication au sein des structures de production : c'est le *"Total Quality Management"* .

Mais arrivé à la fin des années 90, l'importance de la connaissance est devenu tellement cruciale dans les entreprises qu'aucune structure ne peut se passer de protéger et d'améliorer son bagage de connaissance. Par rapport à la philosophie du *"Total Quality Management"* on a mis l'accent sur l'idée qu'il fallait développer une stratégie dite de *"Learning"* c'est à dire privilégier une structure qui permette de susciter le changement plutôt que de s'y adapter. On s'est alors rendu compte que l'apprentissage est naturellement maximisé dans une structure flexible et décentralisé ou un participant au

projet peut évoluer et communiquer avec ses collègues. Cela correspond à une structure coordonnée en forme de toile d'araignée mais sans hiérarchie stricte : *une organisation coopérative en réseau*.

1.2 Les logiciels libres : l'exemple-type d'une oeuvre crée à travers un système de développement coopératif

Ce sont les informaticiens qui ont imaginés et développés les innovations techniques qui ont permis la mise en place des organisations en réseaux. Comme il se doit c'est bien sur eux qui ont eu la primeur de leurs découvertes et qui ont été les premiers à expérimenter ces structures de management en développant des logiciels au sein de structures coopératives. Mais, face à la ré-appropriation de leurs logiciels, les développeurs ont été obligé de se battre et d'imaginer des solutions pour protéger leurs créations sans menacer leur méthode de développement.

1.2.1 Des logiciels libres dès l'origine de l'informatique

Pur produit intellectuel autant que travail d'équipe, les logiciels semblent de toute évidence être l'exemple parfait du projet adapté à une organisation coopérative en réseau.

1.2.1.1 La mise en place de pratiques de développement coopératif grâce au manque d'intérêt des dirigeants

Comme Chandler l'avait montré, les dirigeants ne sont pas capables d'assimiler l'intégralité de la connaissance nécessaire au fonctionnement d'une entreprise, et ce d'autant plus quand cette connaissance connaît un saut qualitatif ainsi que ce fut le cas avec l'apparition de l'informatique, complètement étrangère à la culture de l'époque.

Jusque vers la fin des années 60, les ordinateurs ne possédaient pas de logiciels mais utilisaient des fonctions directement implantées dans leurs composants, les puces électroniques. Les "OS"[2] et les "logiciels" de ces puces n'étaient pas portables d'une machine à une autre et les compagnies informatiques n'attachaient aucune importance à leur code source. En 1969, au sein de Bell Labs (une division d'AT&T), K. Thompson et D. Ritchie commencèrent à créer un OS portable d'une machine à une autre. Il n'existait alors aucun contrôle hiérarchique sur la création des OS parce que la seule compagnie qui aurait eu l'importance suffisante pour le faire était AT&T[3] et qu'elle s'était vu interdire de vendre des OS en 1956 dans le cadre d'un accord antitrust. A ses yeux, le développement des OS n'était donc rien d'autre qu'une nécessité coûteuse pour faire fonctionner ses commutateurs téléphoniques. K. Thompson et D. Ritchie

[2] le système d'exploitation ou Operating System qui permet de faire fonctionner les logiciels sur un ordinateur

[3] la compagnie historique américaine qui possédait le monopole sur les télécommunications

réussirent alors à convaincre AT&T de les laisser donner leur OS à tous ceux qui seraient intéressés.

Ils commencèrent à fournir des bandes[4] de leur OS à différentes universités qui s'en servirent pour faire travailler leurs étudiants en essayant de corriger les bugs contenus dans l'OS d'AT&T. Ils faisaient ensuite remonter les modifications à AT&T qui les incluaient dans la nouvelle version de leur OS et ainsi de suite : les utilisateurs de l'OS étaient considérés comme des développeurs et ils participaient autant à l'élaboration du projet que les ingénieurs de Bell Labs. C'est certainement le premier exemple réussi de développement coopératif en réseau, et il a tellement bien fonctionné que les OS les plus utilisés dans le monde sont toujours bâtis sur les fondations du système créé à l'époque : UNIX.

Mais peu à peu, les propriétaires légaux de cet OS firent valoir leurs droits et, démotivés, ses utilisateurs s'impliquèrent de moins en moins dans son développement.

1.2.1.2 Les problèmes liés à la ré-appropriation des logiciels par les structures hiérarchiques classiques

En 1984, Richard M. Stallman fut confronté à une évolution de licence du driver pour l'imprimante de son laboratoire au MIT. Lui et son équipe avaient développé toute une série de modifications utiles autour du driver de leur imprimante HP en utilisant le code source mais la version suivante du driver fut déclarée propriétaire par HP qui refusa de leur communiquer le nouveau code source. Incapable de modifier le nouveau driver, Stallman ne pouvait plus utiliser les applications qui utilisaient les améliorations qu'il lui avait autrefois apporté...

Après 1984 Unix ne fut plus jamais libre et ceux qui l'avaient utilisé et avaient participé gratuitement à son développement se sentirent complètement trahis. Pour pouvoir continuer à développer un Unix, l'université de Berkeley dut enlever toutes les lignes de code qui dérivaient encore de l'OS original d'AT&T pour créer BSD Unix sans avoir le droit de continuer à profiter d'un OS dont elle avait été un des principaux développeurs.

De son coté, Richard M. Stallman décida de créer un OS et un environnement informatique qui serait conçu comme libre dès l'origine et qui se prêterait donc plus que tout autre à un développement coopératif en réseau : le projet GNU dont GNU/Linux est une branche depuis 1991.

1.2.2 La réussite et le développement des logiciels libres

> "resist the resistance" [Moglen, 1999]

Les plus importantes innovations informatiques des vingt dernières années comme GNU/Linux, Apache (serveur web), Sendmail (serveur mail), Perl, TCP/IP ou Bind (des éléments cruciaux d'internet) ont été construites autour de projets gérés comme des systèmes de développement coopératifs.

Mais l'intuition de Richard M. Stallman fut de ne pas simplement se révolter contre la faveur juridique qui avait évincé la plupart des utilisateurs / développeurs des pre-

[4] on n'utilisait pas encore de support moderne de l'information comme les disquettes ou les disques durs

miers logiciels après que les dirigeants ait pris conscience de la valeur commerciale de ce qui avaient été développé : dans son esprit, le logiciel libre doit être un bien commun de la connaissance. A la manière d'une structure end2end[5] il ne doit pas discriminer l'innovation. Grâce à lui, les utilisateurs de Logiciel Libres ne sont pas les otages des projets Open Source :

- face à un mauvais logiciel : ils peuvent jouer avec le code source et corriger les bugs qu'il contient. Ils sont les seuls à pouvoir supprimer même les plus petits bugs parce que chaque utilisateur est un développeur potentiel et ils peuvent bénéficier de plus de programmeurs qu'une entreprise ne pourrait jamais en proposer.
- face à une stratégie d'entreprise : l'open source offre l'assurance que l'environnement informatique restera neutre face à l'innovation et ne pourra pas se retourner contre des innovations qui rendrait obsolètes les logiciels antérieurs (à l'inverse de ce que Microsoft a toujours réussi à faire en empêchant les utilisateurs de se tourner vers des solutions alternatives) puisque les utilisateurs peuvent toujours créer un *"fork"*[6] et qu'un projet ne disparaît que quand ses utilisateurs disparaissent.
- face à la disparition d'une entreprise : les droits sur ses créations peuvent se voir disséminer entre différentes sociétés et le support / développement de ses logiciels peut être interrompu. C'est ce qui est arrivé à Amiga dont les droits sont répartis aux quatre coins de la planète et empêchent tout redémarrage de son activité.

Les créateurs de logiciels Open Source ont donc repris les structures de l'informatique à ses débuts : ce qu'un auteur crée, il le partage avec ses utilisateurs en leur laissant le plus de libertés possibles à charge pour eux de laisser profiter de leurs améliorations de la même façon. Les projets se développent rapidement et finissent par absorber des centaines de développeurs qui créent des sous-projets capables de s'auto-organiser autour de quelques coordinateurs reconnus pour leur talent à gérer les négociations sur l'évolution du logiciel. Le sociologue Eric S. Raymond compare la création d'un logiciel libre et d'un logiciel commercial à la façon dont se construisent un bazar et une cathédrale en expliquant que le développement sous la forme décentralisée d'un bazar est plus adaptée à la création d'un logiciel car la quantité de bugs découverts est liée à la quantité de développeurs [Raymond, 1998]. Il décrit alors "l'ephemeralization" des logiciels : le coût du développement des logiciels et leur prix ne peut que diminuer à mesure qu'il est de plus en plus facile de les créer [Raymond, 1999][7]. Au niveau juridique, la protection des projets libres passe par la création de la Free Software Foundation et de Richard M. Stallman : la GPL qui utilise paradoxalement le droit de la propriété intellectuelle pour atteindre son objectif..

[5] une structure end2end est une structure semblable à internet où ce n'est pas la structure du réseau qui impose des choix mais plutôt les utilisateurs à ses extrémités : internet ne fait pas de discrimination entre les logiciels qui l'utilisent, en ne favorisant ni l'un, ni l'autre, le réseau laisse le champ libre à l'innovation [Lessig, 1999, chap 14].

[6] un second projet issu du premier à l'initiative de développeurs mécontents

[7] l'"Ephemerisation" est un phénomène décrit par Buckminster Fuller quand une technique devient d'autant plus efficace et moins chère que les ressources physiques investies dans les premiers efforts de développement sont remplacées par de plus en plus de contenu informatif

Chapitre 2

La mise en place d'une structure juridique adéquate de protection des créations coopératives : la Licence Publique Générale

> "We must not believe that if some control is good then more control is better"[Lessig, 2001]

Face à de nouvelles formes de création, il a fallu imaginer une nouvelle manière de les protéger sans pour autant brider la liberté et la souplesse qui faisait leur force. La Licence Publique Générale a été élaborée pour protéger la création coopérative et son objectif libéral. Son succès ne cesse aujourd'hui d'aller grandissant.

2.1 Le caractère contractuel de la Licence Publique Générale

La Licence Publique Générale est un contrat qui organise "la mise en communauté du logiciel" [Lang, 2000, p72] pour qu'il puisse être développé de façon coopérative.

2.1.1 Les raisons de la nature contractuelle de la Licence Publique Générale

Les caractéristiques des créations coopératives, et notamment des logiciels libres, amènent à s'interroger sur la titularité des droits qui pèsent dessus.

2.1.1.1 La titularité des droits du logiciel

La nature des logiciels libres est d'évoluer sans cesse, de s'ouvrir aux suggestions des utilisateurs et des développeurs pour s'améliorer et pour gérer de nouveaux protocoles ou de nouvelles fonctionnalités. Il s'agit donc d'une oeuvre évolutive, fruit du travail et de la réflexion de nombreuses personnes.

Un logiciel est protégeable *du seul fait de sa création* dès qu'il présente un *caractère original*. De nature évolutive, c'est un meuble incorporel qui est protégé par le droit d'auteur au fur et à mesure de sa création. Sauf preuve contraire, *il appartient à celui qui le divulgue*. L 111-1 CPI L 112-4 CPI L 113-1 CPI

Les problèmes apparaissent quand un logiciel est élaboré par plusieurs auteurs. C'est bien sur le cas des logiciels libres où chaque nouvelle version du logiciel a vocation à être une oeuvre dérivée à laquelle ont collaboré plusieurs programmeurs et plusieurs utilisateurs.

2.1.1.2 Le problème posé par la titularité des droits du logiciel dérivé

Un logiciel libre est créé de la façon suivante : un auteur original commence par créer un logiciel auquel il attribue le numéro de version 0.1 (les protocoles de numérotation peuvent être différents en fonction des projets et des autres licences libres) et dont il diffuse le code source, il reçoit ensuite diverses contributions d'utilisateurs / développeurs qu'il intégrera dans la version 0.2 dont il diffuse aussi le code source et ainsi de suite. Les versions dérivées d'un logiciel peuvent recevoir 3 qualifications : oeuvre de collaboration, oeuvre composite ou oeuvre collective.

L'oeuvre de collaboration est définie comme *"une oeuvre à la création de laquelle ont concouru plusieurs personnes physiques"*. On peut difficilement retenir cette qualification dans la mesure où la nouvelle version du logiciel a le statut d'oeuvre dérivée puisqu'elle est dans la dépendance du logiciel antécédent [Lamy Informatique, 2001, n°2977]. L-113-2 CPI

L'oeuvre composite est *"une oeuvre nouvelle à laquelle est incorporée une oeuvre préexistante sans la collaboration de l'auteur de cette dernière"* mais cette qualification n'est pas non plus à retenir puisque l'auteur de l'oeuvre primaire n'a jamais accordé de droits aux développeurs qui ne font qu'adhérer à la licence GPL dont l'objet n'est pas de transférer des droits de propriété du logiciel. L-113-2 CPI

En revanche, la qualification d'oeuvre collective s'impose naturellement. C'est *"l'oeuvre créée sur l'initiative d'une personne physique ou morale qui l'édite, la publie et la divulgue sous sa direction et son nom et dans laquelle la contribution personnelle des divers auteurs participant à son élaboration se fond dans l'ensemble en vue duquel elle est conçue, sans qu'il soit possible d'attribuer à chacun d'eux un droit distinct sur l'ensemble réalisé"*. En pratique, il suffit qu'il soit impossible d'investir chacun des auteurs d'un droit distinct sur l'ensemble pour qualifier l'oeuvre de collective [Desbois, 1978]. La cour de cassation a reconnu la parfaite applicabilité de cette qualification au cas du logiciel en général et elle retient bien comme critères l'initiative d'un auteur originaire suivie de la participation de plusieurs contributeurs (Cass. 1ere civ, 3 juillet 1996, n°92-18.627, JCP éd. G 1997, I, n°657) : nous sommes complètement dans le schéma du développement coopératif d'un logiciel libre. C'est donc L113-2 al 3 CPI

l'auteur de l'oeuvre originaire qui a eu l'initiative du projet qui est investi des droits d'auteurs sur l'oeuvre collective dérivée.

M. Clément Fontaine rejette cette qualification d'oeuvre collective au motif que les développeurs ne reçoivent pas d'ordre du coordinateur du projet [Clément-Fontaine, 1999, n°20] mais, comme l'a très clairement expliqué Eric S. Raymond, une structure de développement coopératif génère ne fonctionne pas sans l'apparition d'un ou des coordinateurs au sens de l'article L 113-2 du CPI [Raymond, 1998].

Mais, telle quelle, cette protection offerte par le statut de l'oeuvre collective génère une insécurité pour les participants à un projet libre : le titulaire des droits peut décider d'utiliser leurs apports créatifs à son profit.

La solution choisie pour sécuriser la relation juridique entre le coordinateur et ses développeurs est d'adopter un contrat protégeant le développement coopératif en utilisant la force de la propriété intellectuelle de l'auteur : la Licence Publique Générale. Comme il est détenteur des droits sur l'oeuvre collective, le créateur du projet pourra décider seul du mode d'exploitation du logiciel et imposer l'usage de cette licence.

2.1.2 Les caractéristiques contractuelles de la Licence Publique Générale

Un contrat doit réunir quatre conditions pour sa validité : *"le consentement de la partie qui s'oblige, la capacité de contracter, un objet certain qui forme la matière de l'engagement, une cause licite dans l'obligation"*. En ce qui concerne la cause du contrat c'est très simplement le droit d'auteur du créateur du logiciel, chaque version du logiciel sera couverte par la GPL pendant 70 ans après la mort de son auteur. Passée cette date, la GPL n'a plus lieu d'être puisque le logiciel n'est plus protégé par le droit d'auteur et qu'il tombe dans le domaine public.

1108 Code Civil

2.1.2.1 La mise à disposition du logiciel

L'objet de la GPL est la mise à disposition du logiciel et non pas, comme on le croit souvent, le transfert ou l'abandon d'un droit de propriété.

La GPL aménage la mise à disposition d'une oeuvre collective mais c'est au titulaire des droits de l'oeuvre de décider de la placer sous l'empire de la GPL. En principe c'est donc *celui qui a divulgué l'oeuvre* mais si il est salarié, c'est à son employeur de prendre cette décision puisque *les droits patrimoniaux lui sont dévolu*s. En revanche, on peut noter que si le logiciel est une oeuvre de commande, c'est encore le créateur qui sera titulaire des droits et qui pourra choisir de le placer sous l'empire de la GPL [Lucas, 2001, n°532].

L 113-1 CPI
L 113-9 CPI

2.1.2.2 Le consentement des parties à la GPL

Le consentement est classiquement l'accord de volonté des parties en vue de créer des effets de droit, la rencontre de ces volontés est la condition de formation du contrat [Cornu].

En matière de propriété littéraire et artistique, le CPI prévoit un important formalisme aux articles L 131-2 et L 131-3 à propos du contrat d'édition et du contrat

de représentation. Les autres opérations relèvent du droit commun des articles 1341 et suivants du Code Civil [Colombet, 1997, n°316 et s.], [Lamy Informatique, 2001, n°856]. C'est donc à bon droit que la GPL stipule à son article 6 que le consentement du licencié est indiqué par le fait de modifier ou de distribuer le programme[1].

2.2 Les conséquences de la Licence Publique Générale

Grâce aux obligations originales qu'elle impose aux parties la GPL a facilité l'expansion continue du logiciel libre et le succès de son modèle de développement coopératif.

2.2.1 Les obligations originales des parties au contrat

La GPL est un contrat dont l'objet correspond aux obligations des parties : le donneur de licence et le licencié.

2.2.1.1 Les obligations du donneur de licence : une remise en jouissance originale et la remise du code source

La GPL confère au donneur de licence l'obligation de procéder à une remise en jouissance originale de la chose, c'est à dire du logiciel [Clément-Fontaine, 1999, n°44 et s.] et, évidemment, de communiquer son code source.

La destination de la chose est la liberté. Le logiciel doit pouvoir être librement utilisé, diffusé et modifié. La liberté ainsi accordé est bien plus grande que celle qu'on peut obtenir par le biais d'un shareware (simple liberté d'essayer gratuité le logiciel) ou d'un freeware (simple gratuité du logiciel) [Guilleux, 1997, p12]. C'est même le point essentiel de la GPL : elle doit déclencher le mécanisme qui permettra l'évolution du logiciel grâce aux interventions des utilisateurs.

Le corollaire évident de cet objectif est la mise à disposition du source du logiciel [Lamy Informatique, 2001, n°2975]. Pour le communiquer, la GPL adopte une définition élargie de la chose. Classiquement, le code source d'un logiciel correspond au programme du logiciel dans un langage humainement compréhensible qui permet son analyse et sa modification et le code objet correspond au produit binaire et inaccessible à la compréhension humaine de la compilation du code source [Lamy Informatique, 2001, n°1016 et s.]. On assimile la plupart du temps le code objet du logiciel au logiciel lui-même mais la GPL fait la différence en liant le code source au code objet à travers son article 4[2]. La jurisprudence française s'est refusée jusqu'à présent à considérer que le code source puisse être un accessoire du logiciel qui devrait obligatoirement être

[1] cf Annexe 1 : En conséquence, par le fait de modifier ou de distribuer le programme (ou un ouvrage quelconque se fondant sur le programme), le concessionnaire indique qu'il accepte la présente licence, et qu'il a la volonté de se conformer à toutes les clauses et dispositions concernant la duplication, la distribution ou la modification du programme ou d'ouvrages se fondant sur ce dernier.

[2] cf Annexe 1 : la distribution du code source et du code objet doit se faire est possible de 3 façons différentes, la GPL prend clairement le temps d'expliquer les raisons de cette obligation et les différentes possibilités offertes pour la mettre en oeuvre

distribué avec lui [Lamy Informatique, 2001, n°1062] mais des voix s'élèvent aujourd'hui pour réclamer que le code source soit considéré comme une facilité essentielle dans le sens où son accès est indispensable pour développer des logiciels basé sur des patches[Reiser, 2002][3].

Quoiqu'il en soit, l'obligation de mettre le code source à la disposition de l'utilisateur est au coeur de la GPL car c'est la seconde condition indispensable à l'évolution du logiciel. La GPL crée une forme de "commons", un élément du Domaine Public disponible avant l'heure[Lessig, 2001]. Par un mécanisme paradoxal expliqué dans son préambule, c'est le droit d'auteur lui-même qui protège le logiciel sous GPL en permettant à son créateur de faire respecter son choix de stratégie de développement. On a pu dire que la GPL était une pratique "subversive" [Lamy Informatique, 2001, n°2973] mais c'est une critique clairement assumée par ses auteurs qui répondent que "l'objectif de la FSF en créant la GPL était bel et bien subversif : réorganiser la façon de créer des logiciels afin de permettre à chacun de comprendre, réparer, améliorer et distribuer les meilleurs logiciels qui soient"[Moglen, 2001].

2.2.1.2 les obligations du licencié : le respect du droit moral de l'auteur et le respect de la libre utilisation du logiciel

Le licencié a deux obligations parallèles à celles du donneur de licence :

- Selon le préambule de la GPL[4], il doit d'abord suivre l'obligation positive de respecter le Droit Moral de l'auteur même si celui-ci est très faible en matière de logiciels [Lucas, 2001, n°651]. En pratique cela ne concerne que le droit de paternité et le droit de divulgation des différentes version du logiciel.
- Le licencié doit ensuite obéir à l'obligation négative de respecter la libre utilisation du logiciel. L'article 10 de la GPL lui impose en effet de demander l'autorisation de l'auteur pour utiliser son logiciel dans un logiciel soumis à une autre licence[5]. L'auteur qui souhaite faciliter l'utilisation de son logiciel dans toutes les sortes d'autres logiciels (freewares, sharewares ou logiciels commerciaux) aura donc intérêt à choisir une autre licence open source (comme la licence BSD qui sert de base à de nombreux protocoles d'internet comme TCP/IP) ou à utiliser la Lesser General Public Licence[6] qui permet d'incorporer plus facilement son programme à des programmes soumis à des conditions de distribution différentes.

[3] la pratique du patch ou rustine consiste à appliquer un correctif à un logiciel pour lui ajouter une fonctionnalité, corriger un bug, etc... Hans Reiser estime que le Code Source des logiciels est une facilité essentielle car elle condition l'accès au marché pour les entreprises qui voudraient développer des patchs pour tel ou tel logiciel

[4] cf Annexe 1 : Si le logiciel est modifié par quelqu'un d'autre puis transmis à des tiers, nous souhaitons que les destinataires sachent que ce qu'ils possèdent n'est pas l'original, de façon que tous problèmes introduits par d'autres ne se traduisent pas par une répercussion négative sur la réputation de l'auteur original.

[5] cf Annexe 1 : Si le concessionnaire souhaite incorporer des parties du programme dans d'autres programmes libres dont les conditions de distribution sont différentes, il devrait écrire à l'auteur pour demander son autorisation

[6] cf Annexe 2

2.2.2 La distribution des logiciels sous licence GPL

Le développement des logiciels libres passe par la possibilité offerte à tous les licenciés de les communiquer à des tiers qui peuvent participer au projet mais la grande force du développement coopératif c'est son aspect extensif.

2.2.2.1 La distribution des logiciels libres auprès des tiers

En accord avec les principes du développement coopératif, tous les licenciés peuvent rediffuser le logiciel mais pour être sur de continuer à protéger le logiciel par la GPL, il leur est interdit de le faire par le biais d'une sous-licence.

L'article 2 de la GPL prévoit ainsi que le licencié doit adresser à tout destinataire du logiciel un exemplaire de la licence GPL ou lui indiquer comment y accéder[7]. Ainsi, chaque fois fois que le licencié distribue le logiciel, modifié ou non, le destinataire reçoit directement du donneur de licence les droits de dupliquer, distribuer et modifier le logiciels selon les termes de la licence [Clément-Fontaine, 1999, n°64]. C'est toujours le donneur de licence initial qui continuera à occuper la place du donneur de licence au fur et à mesure que de nouveaux partenaires s'impliqueront dans le développement coopératif du logiciel [Lamy Informatique, 2001, n°2976-b]. Cette solution très originale permet de garantir la stabilité de l'édifice GNU autour du droit de l'auteur original du logiciel.

2.2.2.2 L'aspect viral de la GPL

Selon l'article 3 de la GPL, le licencié qui souhaite distribuer à son tour le logiciel doit d'abord indiquer les éventuelles modifications qu'il y apportées pour que les autres licenciés de la communauté puissent en bénéficier et que les améliorations apportées puissent être intégrés à la version suivante du logiciel[8].

De cette façon, les logiciels dérivés d'un logiciel GPL sont eux-mêmes des logiciels GPL. Steve Ballmer, PDG de Microsoft n'a pas hésité à qualifier cette particularité d'aspect "viral" de la GPL mais comme le fait remarquer M. Clément-Fontaine, "la boucle est bouclée, le licencié peut devenir donneur de licence et la communauté des utilisateurs du logiciel libre s'agrandit peu à peu" [Clément-Fontaine, 1999, n°69] ce qui est l'objectif toujours recherché par des systèmes de création coopératifs. Il faut noter tout de même que cette extension de la GPL a ses limites indiquées aux articles 3-2 et 3-3 : le développeur a le droit d'exploiter séparément ses modifications et d'utiliser du code GPL dans une oeuvre sous licence différente à condition qu'elle soit raisonnablement indépendante et distincte.

[7] cf Annexe 1 : il appose, d'une manière parfaitement visible et appropriée, sur chaque exemplaire, un avis approprié de droits d'auteur

[8] cf Annexe 1 : Le concessionnaire doit faire en sorte que les fichiers modifiés portent un avis, parfaitement visible, disant que le concessionnaire a modifié les fichiers, avec la date de tout changement.

CONCLUSION

La GPL sert parfaitement son objectif : faciliter le développement coopératif des logiciels. C'est un système qui semble juridiquement fiable à l'issue de cette étude et humainement à la pointe des techniques de management : grâce à l'articulation du droit d'auteur et du contrat, la GPL permet de gérer l'évolution d'un logiciel dans le temps à travers une multiplicité d'auteurs. En privilégiant le partage sur la mise à l'écart elle amène l'idée de "copyleft" plutôt que celle de "copyright" : le contrôle de l'auteur sur la divulgation, la paternité, le respect, la reproduction et la représentation de son oeuvre est destiné à assurer la liberté de l'oeuvre plus que sa réservation.

Si le logiciel libre est développé de façon anarchique au sens ou il est développé sans chef (an-archos), il n'est pas pour autant développé sans organisation juridique puisque toute la structure juridique qui protège son développement se ramène sans cesse à l'auteur original et à son droit sur l'oeuvre.

Mais il reste une menace sur le logiciel libre : le principe du développement coopératif implique la circulation du code source des logiciels et donc la possibilité de captation de ce code source par des tiers qui tenteraient de s'en prévaloir. D'abord il est assez simple d'imaginer qu'une société décide de "voler" le code source d'un logiciel libre pour utiliser les contributions de la communauté au sein de leurs propres logiciels. Heureusement, le développement coopératif repose sur des bases juridiques solides et, alors que la GPL existe depuis 1984, aucun précédent juridique n'est à signaler hormis celui, très récent, de la société Progress Software Corp qui est poursuivie par la FSF pour avoir contrefait des logiciels sous GPL leaders en base de données de la société MySQL AB :
http://linuxtoday.com/news_story.php3?ltsn=2002-02-26-013-20-PR-LL

L'autre menace qui pèse sur les logiciels libres est celle du brevet qui met les tenants d'un système de création coopératif dans la difficulté puisque ce sera à eux de supporter la charge de la preuve de leur antériorité pour faire annuler le dépôt du brevet. Ce problème devient chaque jour plus important alors que la Commission Européenne a résolument choisi cette voie en déposant une proposition de directive entérinant la pratique de l'Office Européen des Brevets de breveter les logiciels :
http://swpat.ffii.org/vreji/papri/eubsa-swpat0202/index.en.html

Cette menace n'est pas théorique et les développeurs d'apt-get et d'urpmi, deux logiciels très importants de mise à jour automatique des stations de travail sous linux ont pu en faire l'amère expérience il n'y a que quelques semaines en découvrant ce brevet :
http://linuxfr.org/2002/02/23/7227,0,1,0,0.php3[9]

Il reste aujourd'hui à espérer que le développement coopératif ne verra pas son expansion brutalement stoppée par l'adoption de principes qui rendraient plus difficiles la communication et le partage de l'information. Mais il faut rester confiant car, après tout, l'expérience de la GPL a montré l'extraordinaire souplesse du droit d'auteur qui, au-delà de la simple protection de la création, permet finalement de protéger la mise à disposition au public d'une oeuvre par son auteur et contribue à faciliter le partage de la connaissance.

[9] cf Annexe 3

soufron@free.fr

Table des matières

Bibliographie

[Dafermos, 2001] Dafermos Georges N. 2001, *Management and Virtual Decentralised Networks : The Linux Project* http://firstmonday.org/issues/issue6_11/dafermos

[Fayol, 1949] Fayol H. 1949, General and Industrial Management

[Chandler, 1962] Chandler A. 1962, Strategy and Structure

[Lessig, 1999] Lessig L. 1999, Code and other laws of cyberspace, Basic Books

[Raymond, 1998] Raymond Eric S. 1998, The Cathedral and the Bazaar http://www.linux-france.org/article/these/cathedrale-bazar/cathedrale-bazar_monoblock.html

[Raymond, 1999] Raymond Eric S. 1999, The Magic Cauldron http://www.linux-france.org/article/these/magic-cauldron/magic-cauldron-fr_monoblock.html

[Bastiat, 1851] Bastiat Frédéric 1851, Harmonies Economiques

[Lang, 2000] Lang B. février 2000, Internet libére les logiciels, La Recherche numéro spécial l'avenir du web

[Moglen, 1999] Moglen Eben 1999, *Anarchism Triumphant : Free Software and the Death of Copyright* http://emoglen.law.columbia.edu/publications/anarchism.html

[Desbois, 1978] Desbois H. 1978, Le droit d'auteur en France, Dalloz

[Lessig, 2001] Lessig Lawrence 2001, the future of ideas, the fate of commons in a connected world, Random House

[Cornu] Cornu Gérard, Vocabulaire Juridique

[Colombet, 1997] Colombet C., Propriété Littéraire et Artistique, Précis Dalloz 1997

[Lamy Informatique, 2001] Lamy Droit de l'Informatique et des Réseaux, 2001

[Clément-Fontaine, 1999] Clément-Fontaine Mélanie, 1999, La Licence Publique Générale, mémoire de DEA sous la direction de Monsieur le Professeur Michel Vivant
http://crao.net/gpl/

[Guilleux, 1997] Guilleux G.A. 1997, Freeware, shareware, crypppleware ; présentation et classification des logiciels en libre copie, droit de l'informatique et des télécoms, 1997/1 p12.

[Reiser, 2002] Reiser : MS Settlement Reflects Deep Failure To Understand Implications of Patching Technology Press Releases,Law and Licences Jan 30th, 2002
http://linuxpr.com/releases/4445.html

[Moglen, 2001] Moglen Eben september 2001, Enforcing the GPL
http://www.gnu.org/philosophy/enforcing-gpl.html

[Lucas, 2001] Lucas André, Devèze Jean, Frayssinet Jean 2001, Droit de l'Informatique et de l'Internet, Thémis Droit Privé PUF