Vol. 16, nº 1

Les justifications philosophiques de la protection du logiciel par le copyright

Virginie Rousseau*

* Travail de l'auteure présenté dans le cadre d'un programme de maîtrise en droit (concentration Droit et technologie) à la Faculté de droit civil de l'Université d'Ottawa.

1. Introduction

À l'ère de «l'environnement numérique»[1], le logiciel est un outil privilégié, un enjeu économique indiscutable et, par conséquent, un bien convoité nécessitant une protection juridique efficace.

Le logiciel, au même titre que l'architecture matérielle du monde numérique, est une pièce maîtresse dans le développement des techniques. La numérisation des textes, des formes, des images et des sons, le transfert et l'exploitation de ces données ne sont pas concevables sans l'intervention de logiciels. Défini comme «tout programme ou ensemble de programmes informatiques visant à réaliser une tâche spécifique», le logiciel incarne l'efficience et la cohérence de l'environnement numérique.

Le logiciel est donc un enjeu économique non négligeable. Depuis les années quatre-vingt et l'introduction de l'ordinateur domestique, le marché du logiciel a connu une véritable expansion. Mais cette croissance s'accompagne d'un affaiblissement, car le logiciel est victime de son propre développement. À titre d'exemple, il est désormais facile de télécharger sur Internet à la fois les logiciels (certains sites sont spécialisés dans la mise à disposition de «sharewares», des logiciels proposés gratuitement au public pour une période d'essai) et les codes permettant de les utiliser sans s'adjoindre des frais obligatoires. Le logiciel nécessite donc une protection juridique efficace et efficiente.

À ses balbutiements, le logiciel fut perçu comme un objet de droit d'auteur, objet traité différemment par les systèmes juridiques de common law, optant pour une version plus utilitaire, et ceux de droit civil, préférant une version plus personnaliste. La différence fondamentale s'inscrit dans les fondements socioéconomiques des

1. A. Lucas, *Droit d'auteur et numérique*, Paris, Litec, 1998, p. 4.

deux systèmes de protection, que nous illustrerons par les systèmes juridiques français et américain. Le droit d'auteur français est un droit de la personnalité tourné vers la protection du créateur. Le *copyright* américain, en revanche, est basé sur une perspective économique protégeant l'investisseur.

L'approche économique du *copyright* met en balance la protection de l'investisseur et l'intérêt du public. L'arrêt de 1785, *Sayre* c. *Moore*[2], illustre ainsi, à travers un prétendu plagiat d'une carte marine, l'équilibre entre «l'intérêt de celui qui a œuvré pour la communauté et l'intérêt de la communauté à profiter du progrès»[3]. En l'espèce, les juges ont donné raison au défendeur qui avait corrigé cette carte et œuvré dans l'intérêt de la communauté. Le *copyright* est donc fondé sur le souci de rémunérer l'investisseur et de stimuler la création. Un équilibre est nécessaire entre l'intérêt privé de l'auteur et l'intérêt du public. La Cour suprême établit, dans la décision *Twentieth Century Music Corp* c. *Aiken*[4], que «The immediate effect of Copyright laws to secure a fair return for an Author's creative labor. But the ultimate aim is, by this incentive, to stimulate artistic creativity for the general public good»[5]. Certains ont avancé que ces intérêts coïncideraient fréquemment, car sans la promesse d'une récompense, l'auteur ne serait pas encouragé et, donc, ne créerait pas[6]. Il s'agit là d'une approche très financière, faisant peu de cas de la spontanéité artistique, mais qui illustre bien la perspective économique américaine.

Le logiciel, en tant qu'œuvre de l'esprit nécessitant de lourds investissements, trouva une protection logique dans le *copyright*. La protection du logiciel par le droit d'auteur français fut en revanche plus contestée, lors de son intégration dans la loi du 11 mars 1957 sur la propriété littéraire et artistique, par la loi du 3 juillet 1985. Le critère d'originalité, décrit comme la marque de la personnalité de l'auteur, était en effet difficilement applicable à ce nouveau type de création. Certaines adaptations ont donc été nécessaires. L'arrêt *Pachot* du 7 mars 1986[7] a ainsi précisé le critère d'originalité en soulignant l'importance de «l'apport intellectuel» de l'auteur, rendu

2. *Sayre* c. *Moore* (1785), 1 East. 361 n, 102 E.R. 139n.
3. A. Lucas, *op. cit.*, note 1, p. 14.
4. *Twentieth Century Music Corp* c. *Aiken*, 422 U.S. 151, 156 (1975).
5. C.H. SHERMAN, H.R. SANDISON, M.D. GUREN, *Computer Software Protection Law*, BNA Books, feuilles mobiles, dernière mise à jour 1991, §201.3.
6. *Ibid*.
7. *Cass. Ass. Plén.7 mars 1986*, 3 arrêts: D. 1986, jurisprudence, p. 405, concl. Cabanes, note Edelman.

possible par «la liberté du concepteur vis-à-vis de certains choix». Malgré les nombreuses contestations, la protection du logiciel par le droit d'auteur est désormais unanimement admise. Les accords de l'Organisation mondiale du commerce sur les aspects des droits de propriété intellectuelle qui touchent au commerce ont, en reprenant la Convention de Berne sur la protection des œuvres littéraires et artistiques, confirmé que le logiciel était protégé en tant qu'œuvre littéraire[8]. Le logiciel dispose donc d'une protection juridique uniforme et de solides fondements textuels.

Malgré cette large acceptation des États de la protection du logiciel sous la coupe des créations littéraires et artistiques, un mouvement né dans les années soixante-dix conteste les droits accordés aux concepteurs. Le mouvement Open Source soutient en effet l'idée d'une liberté factuelle et légale du logiciel[9]: l'utilisation, la distribution et la modification devraient être autorisées sans restriction et le code source devrait être accessible, au même titre que le code objet[10]. Ce mouvement s'oppose donc simplement à l'existence d'un droit de propriété intellectuelle sur le logiciel et, plus spécifiquement, à l'approche économique tendant à rembourser l'investisseur. Ce mouvement est fortement marqué par la logique de curiosité scientifique et de coopération en matière de recherche, que l'on pouvait trouver chez les premiers hackers[11]. L'idéologie Open Source

8. Accord relatif aux aspects des droits de propriété intellectuelle qui touchent au commerce, y compris le commerce des marchandises de contrefaçon (Accord relatif aux ADPIC), art 10(1).
9. M. STRASSER, *A new paradigm in intellectual law? A case against Open Sources*, http://stlr.stanford.edu/STLR/Articles/01_STLR_4/index.htmbin/onldicfre.cgi, dernière consultation le 30/10/2002.
10. Nous expliquerons les termes de «code source» et «code objet» dans les développements de la partie 2.1 du présent texte.
11. Il est intéressant de relever la définition très partiale proposée par le dictionnaire technique et informatique et Internet, à l'adresse http://perso.wanadoo.fr/philippe.decayeux/dico/h.html. Le hacker y est défini comme «avant tout un programmeur informatique. Bidouilleur et curieux, le hacker n'a qu'un seul but, faire évoluer ses connaissances et celles des autres. Le contraire d'un hacker, c'est le Pirate. A ne pas confondre avec Cracker! Le hacker n'a pas d'intention malveillante: c'est un spécialiste, un expert en systèmes informatique et réseau, et en principe à tout ce qui touche de près ou de loin aux couches IP. Ce surdoué en informatique est un enfant terrible des systèmes, technologies et langages de l'info et du net!». Il est ensuite instructif de comparer cette définition avec celle proposée par Richard Stallman sur son site http://www.gnu.org/gnu/thegnuproject.fr.html «L'utilisation du mot «hacker» dans le sens de «qui viole des systèmes de sécurité» est un amalgame instillé par les mass media. Nous autres hackers refusons de reconnaître ce sens, et continuons d'utiliser ce mot dans le sens «qui aime programmer et apprécie de le faire de manière astucieuse et intelligente»».

est diamétralement opposée au monopole des grandes entreprises produisant et distribuant des logiciels. La meilleure illustration de cette lutte est sans doute le conflit qui oppose la société Linux à celle de Microsoft. Bill Gates, furieux, qualifie le mouvement Open Source d'anticapitaliste[12].

Ce mouvement Open Source a désormais une ampleur considérable et trouve un écho auprès des particuliers, des techniciens, des entreprises mais, aussi, des gouvernements. Le gouvernement norvégien a ainsi congédié Microsoft le 12 juillet 2002 et préféré opter pour des logiciels libres, pour des raisons financières[13]. Ce mouvement mérite donc une attention particulière.

Dans cette analyse de la remise en cause de la protection du logiciel par le droit de propriété intellectuelle, nous choisirons de prendre l'exemple du *copyright* américain. Le système américain est sans aucun doute le plus efficace, le plus protecteur du concepteur, et donc, le plus contesté par le milieu du logiciel libre. L'étude abordera, tout d'abord, l'adaptation du *Copyright* au cas particulier du logiciel, en l'envisageant sous l'angle de ses fondements socio-économiques, soit l'intérêt du public par la stimulation de la création et le remboursement de l'investisseur, puis, vérifiera si cette protection est bien justifiée, notamment en la confrontant à la philosophie du mouvement du logiciel libre, et en la comparant aux thèses philosophiques soutenant l'existence de la propriété intellectuelle.

2. Adaptation des fondements du *copyright* au logiciel

L'intégration du logiciel sous la protection du *copyright* est une tâche difficile, du fait de la nature particulièrement technique des programmes informatiques. Un retour aux principes fondamentaux, tels que la dichotomie idée/expression, est donc nécessaire pour déterminer les éléments susceptibles d'être protégés. En effet, distinguer les éléments protégeables des composantes insusceptibles d'appropriation s'avère être une tâche bien délicate, si l'on observe l'exemple du «Reverse Engineering».

12. http://news.zdnet.co.uk/story/0,,t269-s2109446,00.html, dernière consultation le 02/11/2002.
13. http://www.i3c-asso.org/article.php3?id_article=212, dernière consultation le 02/11/2002.

2.1 La dichotomie idée/expression

Le droit américain intègre sous la protection du *copyright*, traditionnellement réservée aux œuvres littéraires et artistiques, certains éléments particulièrement techniques du logiciel. Afin de garantir la cohérence de la notion de *copyright* et s'assurer d'une application uniforme de cette protection au niveau jurisprudentiel, il était nécessaire de respecter scrupuleusement les principes fondamentaux. Les fondements du *copyright*, brillamment synthétisés dans la décision *Twentieth Century Music Corp.* c. *Aiken*[14], consiste en un juste remboursement de l'investisseur, en vue de stimuler la création. La pratique juridique, pour se conformer à ce double objectif et trouver un équilibre entre l'intérêt de l'investisseur et l'intérêt du public, a adopté la dichotomie idée/expression. L'article 102b) du *Copyright Act* dispose que: «In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated or embodied in such work». L'auteur d'une création ne peut ainsi obtenir de *copyright* sur l'idée, mais uniquement sur l'expression de celle-ci[15]. En effet, il faut accorder une protection au créateur pour le récompenser et le stimuler, mais éviter d'accorder un monopole sur les idées, qui anéantirait toute possibilité de création libre ultérieure.

Appliquer cette dichotomie idée/expression au logiciel est une opération délicate du fait de sa nature particulièrement technique. Le logiciel est en effet une structure complexe, mêlant une somme d'éléments. Afin de mieux cerner l'étendue de la protection accordée par la jurisprudence, il est nécessaire d'évoquer rapidement les principales composantes techniques. Les algorithmes, ces procédures visant à résoudre un type donné de problèmes mathématiques[16], sont généralement considérés comme des idées non protégeables au regard de l'article 102b) du *Copyright Act*. En revanche, le code source et le code objet sont facilement considérés comme des expressions, intégrant donc le champ de protection du *copyright*[17]. En effet, ces deux codes, bien que fondamentalement différents, procèdent tous deux d'une création du concepteur impliquant des choix, et ne se limitent pas à la simple incarnation d'une idée fonctionnelle. Le code

14. *Twentieth Century Music Corp* c. *Aiken*, *loc. cit.*, note 4.
15. *Feist Publications Inc.* c. *Rural Tel. Serv. Co.*, 499 U.S. 340, 350 (1991).
16. C.H. SHERMAN, H.R. SANDISON, M.D. GUREN, *op. cit.*, note 5, §203.5.
17. M. STRASSER, *loc. cit.*, note 9.

source désigne l'ensemble des fichiers contenant les lignes d'instructions de langage composant un programme[18], et se base sur un langage compréhensible par l'homme, mêlant écriture et formules mathématiques. Ce code n'est pas directement exécutable et son application nécessite sa traduction en code objet par l'opération de compilation. À l'inverse, le code objet, basé sur un langage binaire incompréhensible pour l'homme, mais conforme au langage utilisé par l'ordinateur qui reçoit les instructions, est directement exécutable.

Bien que la dichotomie idée/expression apparaisse comme une notion simple, son application au logiciel présenta de sérieuses difficultés. Dans la décision *Whelan Associates Inc.* c. *Jaslow Dental Laboratory Inc.*[19], le juge accorda la protection à la structure, la séquence et l'organisation d'un logiciel organisant l'activité d'un laboratoire dentaire. Il déclara que «the purpose or function of a utilitarian work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression of the idea». Se basant sur le fait qu'il existait diverses façons d'organiser un laboratoire dentaire, le juge considéra que le logiciel était une expression et non une idée et accorda donc la protection du *copyright*[20]. Mais la définition trop large de la protection ne permettait pas de différencier entre les différentes composantes du logiciel. Les juges de la cause *Computer Associates International Inc.* c. *Altai Inc.*[21] la critiquèrent, considérant qu'elle ignorait la réalité technique du logiciel qui n'exprimait pas une idée unique, mais une multitude. Cette décision *Altai* marqua donc une rupture d'analyse et l'établissement d'une approche plus protectrice des principes fondamentaux du *copyright*. Elle instaura un mécanisme de filtration des idées, permettant de ne recueillir que les expressions. Cette grille d'analyse, en comparant les éléments relevant du domaine de l'expression relevés dans le logiciel contesté et le logiciel protégé, tend à établir s'il y a similarité et, donc, violation du *copyright*.

Cette analyse est composée de trois étapes[22]. Tout d'abord, le logiciel doit être divisé en différents niveaux d'abstraction, corres-

18. http://perso.wanadoo.fr/philippe.decayeux/dico/c.html, dernière consultation le 4/11/2002.
19. *Associates Inc.* c. *Jaslow Dental Laboratory Inc*., 797 F.2d 1222 (3rd Cir. 1986).
20. M.F. MORGAN, «Trash Talking:The protection of Intellectual Property Rights in Computer Software», (1994) 26 *Ottawa L. Rev*. 425.
21. *Computer Associates International Inc* c. *Altai Inc.*, 982 F.2d 693(2nd Cir. 1992).
22. *Computer Associates International Inc.* c. *Altai Inc.*, précité, note 21.

pondant aux parties le structurant. Ensuite, il faut déterminer et éliminer les éléments non protégeables présents dans chacune des parties, comme les éléments relevant du domaine public ou les expressions seules capables d'incarner une idée. Enfin, il reste à comparer ce noyau à la structure du logiciel protégé par *copyright*.

Cette démarche, bien qu'elle soit décrite scrupuleusement, laisse planer des incertitudes. La nature extrêmement technique du logiciel rend en effet la tâche difficile aux juges peu familiers avec ce type de technologie. La différenciation des idées et des expressions au sein d'un logiciel est en effet tout à fait artificielle. On peut citer à titre d'exemple les interrogations planant sur la qualification à accorder aux éléments de l'interface usager visibles à l'écran d'un ordinateur: s'agit-il d'une expression ou d'une idée?

Ce mécanisme facilite donc l'application de la dichotomie idée/expression au cas particulier des logiciels et garantit un meilleur respect des principes de libre disposition des idées et d'intérêt du public. Malgré ces nombreuses précisions, la particularité du logiciel demeure et certains procédés techniques, tels que le reverse engineering, demandent une attention particulière.

2.2 L'illustration du «Reverse Engineering»[23]

Pour le néophyte, l'ingénierie inverse correspond à l'étude d'un logiciel, en tant que produit fini, en vue de mieux cerner son mode de fonctionnement. Ce mécanisme implique la traduction, par un logiciel spécialisé, d'un code objet, langage binaire directement exécutable par l'ordinateur, en un code source, langage plus élaboré et compréhensible par les techniciens[24]. Les termes «désassemblage» et «décompilation» sont fréquemment présentés comme des synonymes de l'ingénierie inverse. Toutefois, Sunny Handa précise que la décompilation correspond davantage à la traduction du code objet en un haut niveau de langage[25], tandis que le désassemblage mène à un simple langage «machine» compréhensible par les spécialistes informatiques. Les logiciels de désassemblage, caractérisés par une plus grande flexibilité technique, sont donc employés dans la majorité des opérations de «reverse engineering».

---

23. L'ingénierie inverse («Reverse Engineering») est la traduction choisie par Denis BORGES BARBOSA, dans son article «Logiciel et droit d'auteur: un mariage de déraison», (1998) 101 *Le Droit d'auteur* 205, 233.
24. S. HANDA, «Reverse Engineering Computer Programs Under Canadian Copyright Law», (1995) 40 *McGill L.J.* 621.
25. Le haut niveau de langage correspond à un langage commun compréhensible par l'homme.

Les interrogations soulevées par ce processus d'analyse et de recomposition des logiciels illustre parfaitement les doutes suscités par de perpétuelles avancées techniques. La doctrine et la jurisprudence s'interrogent en effet sur une éventuelle violation, par l'opération de désassemblage, des droits conférés par le *copyright*. Le «Reverse Engineering» pose en réalité des doutes à différents stades du traitement du logiciel protégé. Trois violations principales ont été alléguées par les propriétaires de logiciels protégés: la copie constituée par la mise en mémoire du code objet protégé, en vue de procéder au désassemblage; le code source obtenu par cette opération; le nouveau code objet créé après analyse des informations recueillies. D'après Michael F. Morgan[26], le code source obtenu n'engendre pas, par lui-même, une violation du *copyright*, car les contraintes purement techniques empêchent d'obtenir une copie exacte du code source initial. Cette allégation ne retiendra donc pas notre attention.

Ensuite, l'affirmation qu'un code objet créé à partir d'une opération d'ingénierie inverse constitue en lui-même une violation du *copyright* du logiciel inital est largement critiquable. Condamner d'emblée ce logiciel tendrait à interdire l'étude scientifique des programmes existants et s'opposerait donc directement aux principes de stimulation de la recherche et de la créativité et, simultanément, à l'intérêt du public. Ce nouveau logiciel relève donc de l'analyse en trois parties décrites par l'arrêt *Computer Associates International Inc.* c. *Altai Inc.*[27]. La violation ne sera constituée que si le noyau final d'expression copie la structure du logiciel initial. Dans cette logique, la décision *E.F. Johnson Co.* c. *Uniden Corp. Of America*[28], sans étudier si le «Reverse Engineering» constituait une transgression du *copyright*, établit qu'il n'y aurait pas eu de violation si le défendeur avait uniquement utilisé des parties fonctionnelles du logiciel, ne pouvant être produites par une autre voie. Cette tendance jurisprudentielle s'inscrit donc parfaitement dans le respect des principes fondamentaux du *copyright*.

Notre attention doit désormais se porter sur l'éventuelle violation du *copyright*, constituée par la copie effectuée lors de l'opération de désassemblage. L'opération de «reverse engineering» ne peut en effet techniquement s'effectuer sans la mise en mémoire du code, et

26. M.F. Morgan, *loc. cit.*, note 20.
27. *Computer Associates International Inc.* c. *Altai Inc.*, *loc.cit.*, note 21.
28. *E.F. Jonhson Co.* c. *Uniden Corp. of America*, 623 F. Supp. 1485 (D.C. Minn. 1985).

donc, sa copie. Cette question fut abordée, à quelques mois d'intervalle, sous l'angle du «fair use» dans deux décisions. Dans l'affaire *Atari Games Corp* c. *Nintendo of America Inc.*[29], la société Nintendo arguait une violation de son *copyright* sur son 10 NES, un programme encastré dans le disque dur de sa console et qui traitait les informations contenues dans les cartouches de jeu. Elle s'assurait ainsi que les concurrents ne commercialiseraient pas de jeux compatibles avec la console sans lui avoir préalablement payé une licence. Atari, alléguant faussement qu'il était poursuivi en justice et n'utiliserait la copie du code de Nintendo que pour les fins de sa défense, en obtint un exemplaire du Bureau du droit d'auteur américain. Compte tenu de ce comportement frauduleux, la cour refusa d'accueillir les arguments d'Atari fondés sur l'usage équitable. Néanmoins, elle note que les tribunaux doivent adapter l'exception de l'usage équitable aux innovations technologiques[30]. Un individu ne peut observer, encore moins comprendre, un logiciel sans une opération d'ingénierie inverse. L'ingénierie inverse effectuée pour découvrir l'idée non protégée du programme d'ordinateur est donc considérée comme une utilisation équitable[31]. En revanche, si le but ultime du désassemblage vise à profiter de copies d'expression protégées, cette exception ne peut s'appliquer.

La décision *E.g. Sega Enterprises Ltd.* c. *Accolade Inc*.[32], basée sur des faits similaires[33], étudie plus précisément l'exception du «fair use». L'article 107 du *Copyright Act* dispose que:

> The fair use of a copyrighted work, including such use by reproduction in copies or phonorecords or by any other means specified by [106, 106A], for purposes such as criticism, comment, news reporting, teaching (including multiple copies for classroom use), scholarship, or research, is not an infringement of copyright.

29. *Atari Games Corp* v. *Nintendo of America Inc.*, 975F. 2d 832 (Fed. Cir. 1992).
30. *Ibid.*
31. M. B. NIMMER, D. NIMMER, *Nimmer on copyright*, New York, Mathew Bender & Co. Inc., 2002, p. 13-230.
32. *E.g. Sega Enterprises Ltd* c. *Accolade Inc.*, 977 F.2d 1510 (9th Cir. 1992).
33. Sega, souhaitant lutter contre la piraterie, met en place sur sa nouvelle console Genesis 3 un nouveau système de sécurité (Trademark Security System) qui recherche la compatibilité de cartouche de jeu introduite. Accolade perce le code de ce système par une opération de «reverse engineering» et commercialise un jeu compatible avec la console Sega. Sega intente une action en justice et obtient une injonction auprès des premiers juges.

> (1) the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes;
>
> (2) the nature of the copyright work;
>
> (3) the amount and substantiality of the portion used in relation to the copyrighted work as whole; and
>
> (4) the effect of the use upon the potential market for or a value of copyrighted work.
>
> The fact that a work is unpublished shall not itself bar a finding of fair use if such finding is made upon consideration of all the above factors.

Quatre facteurs d'analyse sont à prendre en considération. En premier lieu, le critère du but et du caractère de l'utilisation ne plaidait pas spontanément en faveur d'Accolade, qui souhaitait commercialiser des jeux compatibles avec les consoles Sega. La Cour jugea que la recherche scientifique était le but premier et que le but commercial n'était que secondaire. En second lieu, la Cour observe que la nature de l'œuvre protégée par le *copyright* est avant tout utilitaire et caractérisée par l'efficience. Accorder une protection sur ces éléments non protégeables reviendrait à accorder à Sega un monopole. La Cour, qui souhaite rendre un jugement qui pourra stimuler la création et préserver la liberté des idées, préfère limiter la portée du droit d'auteur. En troisième lieu, le critère de la portion de l'œuvre utilisée n'est pas considéré comme un obstacle à l'usage équitable, même si en l'espèce la totalité du code objet a été reproduite. En dernier lieu, le critère de l'effet de l'utilisation sur le potentiel commercial de l'œuvre n'était pas décisif dans cette affaire. La Cour préfère privilégier la stimulation de la créativité et l'existence d'une saine concurrence, plutôt que se concentrer sur l'affectation prévisible du marché[34].

34. Des décisions s'inscrivant dans la même logique que celle établie par les affaires *Sega* et *Attari* ont été rendues. Ainsi, il a été jugé qu'une opération de téléchargement d'un logiciel d'entretien d'un programme informatique ne pouvait être considéré comme un usage équitable. En revanche, l'ingénierie inverse pour offrir des produits concurrentiels pouvait s'inscrire dans une opération d'usage équitable. Se référer à M. B. NIMMER, D. NIMMER, *loc. cit.*, note 31, p. 13-233.

Le «Reverse Engineering», même s'il n'est pas exclu du champ des violations par une loi particulière, est donc admis par la jurisprudence qui se fonde avant tout sur les principes fondamentaux du *copyright*.

La protection du logiciel par le *copyright* s'organise avec cohérence par l'application rigoureuse des principes fondateurs de cette protection. Mais cette protection est-elle véritablement adaptée à la réalité sociologique et technique? Cette protection du logiciel trouve-t-elle véritablement une justification et des fondements dans la philosophie de la propriété incorporelle?

3. Mise à l'épreuve et justifications de cette protection

La protection du logiciel par le *copyright* est largement critiquée, notamment par le mouvement Open Source. Les partisans du logiciel libre avancent notamment que cette protection ne respecte pas les principes fondamentaux du *copyright*. Suite à de telles critiques, un retour aux théories philosophiques fondant la propriété intellectuelle semble nécessaire pour justifier la protection juridique du logiciel, et plus précisément, la protection de son créateur.

3.1 La confrontation au mouvement du logiciel

Le mouvement Open Source s'est développé à l'ouverture du marché des programmes informatiques, lorsque le partage des logiciels par les membres de la communauté scientifique a fait place à la distribution de copies non exécutables, sous la seule condition de non divulgation. Richard Stallman[35] a alors débuté une longue marche contre la protection du logiciel par le *copyright* et le droit d'auteur, en arguant que «la première étape de l'utilisation d'un ordinateur était de promettre de ne pas aider son prochain»[36] et en dénonçant l'interdiction de toute communauté coopérative. Ce mouvement de contestation prit forme dans un projet de création d'un système d'exploitation libre, compatible avec Unix, le projet GNU. Richard Stallman devint ainsi le chef de file de ce mouvement de «Free Software», animé par cette conviction que l'entraide et le travail en communauté devaient prévaloir sur les monopoles artificiels accordés par la protection du *copyright*.

35. Richard Stallman est cité par Mathias Strasser, comme le père spirituel du mouvement Open Source. M. STRASSER, *loc. cit.*, note 9.
36. http://www.gnu.org/gnu/thegnuproject.fr.html, dernière consultation le 08/11/2002.

Le mouvement Open Source plaide clairement pour un abandon de la protection organisée par le *copyright* et son remplacement par une liberté tant juridique que factuelle[37]. La liberté juridique consisterait à admettre les libres utilisations et modifications des logiciels, les redistributions de copies identiques ou modifiées, et à refuser les divers droits limitant leur usage et exploitation par le public, tels que les licences. Parallèlement, la liberté factuelle se traduirait par une ouverture au public du code source, habituellement jalousement gardé par le développeur du logiciel et, parallèlement, l'offre de l'étudier et le modifier. Chaque individu pourrait donc prendre part au développement du logiciel et profiter gratuitement des avancées scientifiques. Le mouvement du logiciel libre avance en effet que la protection actuelle n'est pas adaptée à la réalité technique du logiciel, pire, qu'elle n'est pas justifiée au regard des fondements du *copyright*.

En premier lieu, les adeptes de l'Open Source affirment que les notions de création et d'originalité fondant la protection ne sont pas parfaitement adaptées au développement des logiciels. En effet, la création d'un nouveau logiciel s'apparente plus à une réadaptation qu'à une création, car le développeur réorganise davantage des portions de programmes déjà existants, qu'il n'en crée[38]. L'originalité du nouveau logiciel est donc largement critiquable.

En second lieu, le mouvement Open Source soutient que le *copyright* n'assure pas une stimulation optimale de l'activité créatrice. En cas de virus informatique, le logiciel libre mobilise en effet l'attention de toute une communauté de développeurs. Tandis que la version «propriétaire»[39] d'un logiciel ne relève que de la compétence des techniciens de l'entreprise, ayant seuls accès au code source. Le mouvement Open Source est ainsi caractérisé par son efficacité et son enthousiasme à solutionner les problèmes et créer des versions de logiciels plus adaptées et efficientes.

En troisième lieu, le mouvement Open Source met en doute l'existence d'une véritable protection de l'intérêt du public par le *copyright*. Cette allégation est illustrée par quatre exemples, avec une appréciation de la notion d'intérêt du public d'importance croissante. Tout d'abord, l'intérêt du simple particulier, souhaitant con-

37. M. STRASSER, *loc. cit.*, note 9.
38. *Ibid*.
39. Expression empruntée à Richard Stallman. Voir http://www.gnu.org/gnu/thegnuproject.fr.html.

necter son vieil ordinateur avec son imprimante flambant neuve, est mieux protégé par un logiciel libre, que par une version «propriétaire»[40]. En effet, si l'ordinateur n'est pas formaté pour reconnaître un nouveau type d'imprimante, l'accès au code source devient indispensable à la résolution de ce problème technique. Le logiciel libre permet donc une adaptation à chaque situation particulière, protège la liberté d'opter pour le matériel informatique de son choix et contribue au maintien d'une saine concurrence.

Le mouvement Open Source œuvre ensuite dans l'intérêt économique du public. Contrairement aux allégations des firmes propriétaires et distributrices de programmes informatiques[41], le logiciel libre n'anéantirait pas *de facto* le marché du logiciel. La demande ne cesserait pas sous prétexte des possibilités de copies libres. Le développement de différents modèles de distribution peut en effet générer un marché important. Au-delà de cette viabilité économique, le logiciel libre aiguiserait une saine concurrence et anéantirait les monopoles artificiels accordés par le *copyright*. L'Open Source présente donc une alternative intéressante pour tout particulier, toute entreprise ou tout État, notamment pour des États en voie de développement. Ces pays n'auraient pas à payer le prix d'un monopole comme celui de Microsoft[42], et la flexibilité du logiciel libre contribuerait à «assurer la protection des cultures locales, le multilinguisme, le développement et la conservation de l'information»[43].

D'après le mouvement Open Source, l'ouverture des codes sources engendrerait en elle-même une augmentation de la somme de connaissances communes de l'humanité. L'augmentation des connaissances communes, par sa promesse intrinsèque d'amélioration de la condition humaine, s'inscrit évidemment dans l'intérêt du public.

40. M. STRASSER, *loc. cit.*, note 9.
41. Le «free software» conduit par Richard Stallman se confronta à de nombreuses critiques basées sur la viabilité économique du mouvement. Elle est essentiellement due à la confusion existant en anglais autour du terme «free». Il ne désigne pas ici la gratuité, mais bien la liberté. Toutefois, certains membres ont préféré le terme d'Open Source et progressivement rebaptisé le mouvement.
42. Suite à une proposition d'Avril, un groupe de travail a remis une demande de classement des logiciels libres au rang du patrimoine de l'humanité par l'UNESCO, à l'occasion des journées d'échange et de promotion autour du logiciel libre, le 12 juillet dernier. http://www.uzine.net/breve1007.html, dernière consultation le 08/11/2002.
43. http://www.uzine.net/breve1007.html, dernière consultation le 08/11/2002.

Enfin, le logiciel libre se pose en fervent défenseur de la liberté de l'homme. Lawrence Lessig affirme en effet que le code, incarnant une des quatre contraintes régissant les comportements sociaux[44], au même titre que la loi, la norme sociale et le prix, incarne désormais la loi, dans le contexte du logiciel. Les développeurs, en codant des systèmes de sécurité dans des cartouches de jeux notamment, s'arrogent un contrôle sur l'utilisation de leur création et une protection plus importante que celle initialement prévue par le *copyright*. Partant du principe que le code incarne la loi des programmes informatiques, Lessig s'interroge sur la transparence des logiciels dont le code source est jalousement gardé. Ces programmes dissimulent en effet l'ensemble de ses procédures et apparaissent donc comme un ennemi potentiel de la liberté de l'homme. L'intérêt du public semble, par conséquent, se trouver davantage dans la solution de l'Open Source que dans la protection du *copyright*.

L'Open Source, au moyen d'arguments pertinents, met à l'épreuve la justification de la protection du logiciel par le *copyright*. Cette protection n'apparaît donc plus aussi évidente que peut l'être un régime juridique établi. Il est nécessaire de revenir aux théories philosophiques proposées pour fonder le *copyright*, pour déterminer la pertinence de sa justification.

3.2 La justification par les thèses fondatrices de la propriété intellectuelle

Le logiciel est sous la protection du système de propriété intellectuelle. L'existence de cette propriété intellectuelle, distincte de la propriété dite tangible, se justifie par la notion d'exclusivité des droits. L'objet du droit de propriété intellectuelle peut en effet être partagé entre un nombre important de sujets de droit sans que sa valeur en soit diminuée, tandis que l'objet du droit de propriété dite tangible s'en trouverait altéré[45]. La propriété intellectuelle est fondée sur différentes théories, s'apparentant soit aux droits économiques, soit aux droits moraux[46]. Ces deux fondements basent les deux systèmes de protection occidentaux, le droit d'auteur et le *copyright*, et justifient leurs différences[47].

---

44. L. LESSIG, «The Law of the Horse: What Cyberlaw might teach», (1999) 113 *Harvard Law Review* 501.
45. M. STRASSER, *loc. cit.*, note 9.
46. S. HANDA, *Copyright Law in Canada*, Markham, Butterworths, 2002, p. 66.
47. *Supra*, p. 4.

Le système américain se fonde sur une conception essentiellement économique de la propriété intellectuelle, même si la ratification de la Convention de Berne a impliqué l'acceptation de quelques notions de droit moral. La clause 8 de la Constitution américaine donne en effet pouvoir au Congrès de promouvoir le progrès des sciences et des Arts utiles, en accordant un droit exclusif aux auteurs et inventeurs sur leurs œuvres et découvertes[48]. Par conséquent, l'étude des justifications philosophiques de la protection du logiciel par le *copyright* s'attachera aux arguments de la théorie économique, mais aussi particulièrement, à la théorie du labeur et de la récompense («labor and dessert»).

La théorie du labeur et de la récompense, soutenue à l'origine par John Locke, établit que le droit de propriété sur un bien revient à l'homme qui l'a créé à la sueur de son front[49]. Un individu n'a pas de droit, excepté les droits sur son propre corps et son propre travail. Cette théorie est fondée sur la loi positive, fortement marquée par l'existence de Dieu. Dieu a accordé aux hommes la maîtrise de leur propre corps et, ainsi, le choix de travailler. Le produit du travail est donc analysé comme une extension du corps humain et relève, selon la volonté de Dieu, de la maîtrise de son créateur[50]. Dès que le travail intervient, un droit de propriété apparaît et la libre disposition du public est condamnée. Locke a toutefois intégré une double condition[51]: le bien ne peut échapper au domaine public que s'il est suffisant et si l'application commerciale demeure à la disposition de tous. Cette théorie se traduit, dans le domaine de la propriété intellectuelle, en accordant au créateur un droit exclusif sur son œuvre, tout en enrichissant la connaissance commune de l'humanité. Elle a longtemps influencé la protection du *copyright*. La théorie du «sweat-of-the-brow» établissait que la dépense de temps et d'efforts était une raison suffisante pour obtenir la protection du *copyright*[52]. Bien que la jurisprudence considère désormais que le simple travail n'est plus pertinent pour justifier la propriété intellectuelle, la théorie a tout de même marqué les fondements de cette protection[53].

48. U.S. CONST., Art. I, § 8, cl. 8.
49. S. HANDA, *ibid*, note 46, p. 87.
50. J. LOCKE, *Second Treatise of Government*, Cambridge, Hackett publishing Co, 1980, [première publication en 1690].
51. G.P. MILLER, «Economic Efficiency and the Lockean Proviso», (1987) 10 *Harv. J.L. & Pub. Pol'y* 405.
52. *Feist Publications Inc.* c. *Rural Tel. Serv. Co.*, 499 U.S. 340, 350 (1991), *loc.cit.*, note 15.
53. M. STRASSER, *loc. cit.*, note 9.

Cette théorie est pourtant facilement critiquable. Locke n'a en effet jamais précisé que son raisonnement s'appliquait aux biens immatériels et à la propriété intellectuelle. L'acte d'appropriation et les conditions de suffisance et de non-gaspillage, tels qu'ils sont décrits par l'auteur, ne semblent pas adaptés à cette catégorie particulière de biens. Considérer l'acte de travail comme un acte d'identification de l'invention ou de l'expression de l'idée, au sens commun de biens incorporels, apparaît en effet très artificiel[54]. Il paraît de même peu probable d'épuiser le stock d'idées susceptibles d'être exploitées. Le raisonnement de Locke semble donc être un argument fragile pour s'opposer aux revendications du mouvement Open Source. Il est donc nécessaire de rechercher des justifications à la protection du logiciel par le *copyright*, au sein d'autres théories.

La théorie économique s'appuie sur le principe que chaque individu est un acteur économique rationnel qui cherche à maximiser la richesse et que les normes juridiques sont entièrement dictées par ce souci d'efficience. Richard Posner[55] fonde cet objectif d'optimisation des richesses sur la rareté des ressources et leur exclusivité d'utilisation. Seule une libre et saine concurrence sur le marché, où se rencontrent l'offre et la demande attachées à ces biens tangibles, assure la meilleure optimisation des richesses. Mais les droits de propriété intellectuelle n'étant pas caractérisés par cette exclusivité, une adaptation de la théorie s'avère profondément nécessaire. La loi doit mettre en place des mécanismes de justes récompenses, stimulant la création de l'auteur. La faiblesse de l'œuvre consiste dans la possibilité de la dupliquer sans altérer sa structure et sans engager de frais substantiels. La solution consiste donc à maîtriser les copies exécutées de l'œuvre, sans condamner la concurrence. En pratique, l'application de la théorie utilitaire aux droits de propriété intellectuelle se traduit par un monopole accordé à l'auteur, limité à l'expression d'une idée et à une période prédéterminée[56]. À l'expiration de ce monopole, les droits tombent dans le domaine public et l'œuvre peut donc être reproduite sans frais. La théorie utilitaire défend donc l'idée d'un équilibre nécessaire entre le remboursement de l'auteur et la stimulation de la créativité. Or, cet équilibre s'avère être la base même du *copyright*. La protection du

54. P. DRAHOS, *A philosophy of Intellectual Property*, Darmouth, Aldershot, 1996, p. 49.
55. R.A. POSNER, «The Economic Approach of Law», (1975) 53 *Tex. L. Rev*. 758.
56. R.E. MEINERS et R.J. STAFF, «Patents, *Copyright*s, and Trademarks: Property or Monopoly», (1990) 13 *Harv. J.L. & Pub. Pol'y* 911, 913.

logiciel par le *copyright* semble donc largement justifiée par la théorie utilitaire. Soutenue par des principes généraux, cette protection prend davantage de profondeur et apparaît moins artificielle ou inadaptée.

Toutefois, la théorie utilitaire semble perdre de sa pertinence lorsqu'on la confronte aux arguments du logiciel libre. Les théories utilitaires et Open Source s'accordent en effet sur l'affirmation qu'une saine concurrence est la condition indispensable à la stimulation de la création. Mais elles diffèrent sur leurs approches de cette libre et saine concurrence. Les partisans soutiennent ainsi que le logiciel libre, en abolissant les monopoles artificiels créés par les brevets et le *copyright*, aiguise l'équilibre naturel du jeu de l'offre et de la demande et stimule les activités inventives et économiques. Au contraire, la théorie utilitaire affirme que seule un juste remboursement de l'auteur est susceptible d'encourager la créativité, développer une activité économique viable et établir une saine concurrence. Ces deux raisonnements semblent fondés. Seule l'abolition de la protection des logiciels par le *copyright* et une observation de vingt ans permettraient peut-être d'établir avec certitude l'exactitude de l'une ou l'autre de ces théories [...].

4. Conclusion

Malgré les critiques virulentes du mouvement Open Source, la protection du logiciel par la propriété intellectuelle se justifie au regard de la théorie utilitaire. La protection particulière par le *copyright* est, elle aussi, fondée, car elle instaure un juste équilibre entre l'intérêt du créateur, en prévoyant une juste compensation par l'octroi d'un monopole temporaire limité à l'expression de l'idée, et l'intérêt du public, en assurant la liberté des idées et la stimulation de la création. Bien que la protection du logiciel par la propriété intellectuelle apparaisse artificielle, du fait de son assimilation à toute autre création littéraire et artistique, elle s'appuie sur des principes solides visant la liberté de la création. Les partisans affirment pourtant que cette protection ne contribue pas suffisamment à cette liberté des idées et s'avère donc inadaptée à la nature technique du logiciel.

Mais une pire menace plane déjà sur le logiciel et la liberté de ses développeurs: le brevet. Cette protection accorderait un monopole sur des séquences entières d'algorithmes et anéantirait en

grande partie les possibilités de développement libre de logiciels. Cette menace est pourtant déjà une réalité aux États-Unis et au Japon et est au centre d'un débat passionné en Europe. Cette protection par le brevet, largement inspirée par des lobbys de grosses entreprises, relève plus de la volonté d'assurer une rentabilité économique que du souci de protéger la liberté des idées. Mais tout excès appelle la réponse d'un régulateur. Reste à choisir entre le monopole de Microsoft et son activité de lobbying, et la communauté Open Source et son idéologie coopérative.