Le logiciel libre : gérer collectivement les évolutions d'une technologie

Nicolas JULLIEN
IMT Atlantique, LEGO-Marsouin

Robert VISEUR
Université de Mons

Jean-Benoît ZIMMERMANN
GREQAM-AMSE (Aix-Marseille Université, CNRS et EHESS)

Dans cet article, nous analysons l'émergence du logiciel libre dans le contexte de la protection intellectuelle du logiciel. Le logiciel libre ne constitue pas une négation du principe de propriété intellectuelle. Alors que la plupart des modèles classiques de propriété intellectuelle ont comme source de revenus la monétisation de l'accès à une technologie (le « logiciel stock »), l'idée du logiciel libre est d'utiliser le droit d'auteur pour organiser, *via* des licences spécifiques, la gestion de l'évolution de la technologie et de son interopérabilité (le « logiciel flux »), qui sont tout aussi importantes pour l'utilisateur que le stock.

Introduction

Comme expliqué par Jullien et Zimmermann (2006), la protection intellectuelle des logiciels est relativement récente, et s'explique par des évolutions technologiques, et ensuite par des nécessités concurrentielles. Au début de l'industrie informatique, cette question ne se posait pas dans la mesure où le logiciel apparaissait comme porteur d'une logique de programme, substitut de la logique câblée du matériel, qui permettait une plus grande flexibilité aux machines et leur donnait la capacité de traiter une pluralité de tâches sur la même architecture. Avec l'émergence d'une distinction claire entre le logiciel système et le logiciel d'application, l'ordinateur est devenu une machine universelle capable d'être dédiée à n'importe quelle application, dans les seules limites de ses propres performances de traitement. Mais tant que les logiciels d'application étaient développés et fournis par les fabricants d'ordinateurs eux-mêmes, les programmes informatiques ont été considérés comme des produits communs dont la fourniture faisait partie intégrante des arguments de marketing pour vendre un système donné. Pour permettre à des entreprises de vendre des services, notamment de développement logiciel, il a fallu séparer les coûts de développement (et de vente) du matériel de ceux du logiciel, notamment chez l'acteur dominant du marché, IBM.

Ces dimensions ont donné aux logiciels une position de bien marchand, et engendré la nécessité de déterminer un cadre pour la reconnaissance et la protection de leur propriété intellectuelle. Les premières études consacrées à cette question en Europe ainsi qu'aux États-Unis dans les années 1970 ont reconnu qu'aucun des cadres de protection existants ne pouvait être considéré comme totalement satisfaisant. Mais elles recommandaient d'éviter les longs délais nécessaires à la conception d'un nouveau cadre et à son adoption générale au niveau international, ainsi que le risque d'une obsolescence rapide

résultant d'une évolution technologique rapide et profonde (voir OTA, 1992). Pour ces raisons, la plupart des pays développés ont décidé d'adopter la loi sur le droit d'auteur comme référence commune pour la protection de la propriété intellectuelle des logiciels, avec plusieurs variantes résultant des contextes juridiques nationaux. La production d'un code logiciel est ainsi protégée, et la reproduction comme l'usage *a priori* interdits, soumis à licence (autorisation) par l'auteur du logiciel.

Pourtant, depuis le début des années 2000, le modèle alternatif du logiciel libre prend une importance croissante. Son principe essentiel est d'imposer à ses adoptants, grâce à des licences d'utilisation particulières, la divulgation du code source des programmes concernés et de toute amélioration ultérieure s'ils les diffusent, ainsi que la libre circulation du code à la seule condition de maintenir son caractère « ouvert ».

De fait, l'approche du logiciel libre ne représente pas un déni de la propriété intellectuelle, mais une nouvelle façon de la gérer. Les auteurs ne renoncent pas à leurs droits, mais à la seule rente de monopole que ces droits autoriseraient dans un régime de droit d'auteur. Cela n'exclut pas une éventuelle commercialisation de ces programmes ou, plus exactement, nous le verrons, de services comprenant ces programmes, et ne limite pas les « logiciels libres » à une sphère non commerciale. Reste une situation paradoxale : pourquoi des acteurs économiques renoncent-ils à des éléments de contrôle permis par la propriété intellectuelle ?

Nous reviendrons d'abord sur les problèmes que pose le système de protection du logiciel, qui est un bien très particulier, car mélange de produit (objet fini) et de service (de gestion de l'adaptation et de l'évolution de cet objet). C'est précisément cette dualité qui est à l'origine et l'intérêt du logiciel libre. Nous conclurons enfin sur l'idée que le « libre » est un système de production intellectuelle visant particulièrement à organiser l'évolution continue de la demande et de l'innovation, ce que Teece *et al.* (1997) appellent les « capacités dynamiques ».

Les limites du droit d'auteur

La protection par le droit d'auteur du logiciel s'est avérée peu satisfaisante pour les utilisateurs et pour la dynamique d'innovation, en raison de la nature très spécifique du bien logiciel et de ses conditions de production et d'utilisation.

La plupart des éditeurs commercialisent leurs produits logiciels sous la seule forme de programmes exécutables. Ils ne révèlent généralement pas le « code source » des programmes, c'est-à-dire l'expression explicite de l'architecture, des procédures et des algorithmes du programme. Cette occultation apparaît contradictoire avec les objectifs de protection de la propriété intellectuelle qui, dans leurs fondements, consistent à accorder un monopole temporaire à l'inventeur pour l'exploitation industrielle et commerciale de son invention en échange duquel l'inventeur est tenu de dévoiler les principes de son invention, favorisant ainsi la diffusion des savoirs techniques dans le tissu industriel (Vivant, 1993). Elle rend aussi l'usage concret de la technologie plus difficile.

En effet, une solution informatique est un mix de produit et de service (Horn, 2004 ; Cusumano, 2004). Si avec l'arrivée des progiciels dans les années 1980, la partie produit est devenue visible, la partie service est en fait restée majoritaire en termes de valeur (Campbell-Kelly et Garcia-Swartz, 2015), surtout chez les clients professionnels, notamment les grandes organisations (Yost, 2017). Illustré par le terme « coût total de possession » (TCO en anglais), l'ensemble du cycle de vie d'une solution est la combinaison de cinq types de coûts (Shaikh et Cornford, 2011) :

- coût d'exploration (définition du besoin, recherche, évaluation et preuve de concept, ou "POC") ;

- coût d'acquisition (prix de la licence, adaptation aux besoins et intégration technologique) ;
- coût d'intégration (dans les usages ; migration, formation et processus) ;
- coût d'usage (maintenance, mise à jour de la technologie) ;
- et coût de retrait ou de sortie.

Dans le modèle classique, cet écosystème est organisé autour d'un éditeur-distributeur, dont le métier est de « gérer une technologie (sur le long terme) » (Cusumano, 2004, p. 4 et suivantes) et de distribuer ce produit, mais pas seulement de fournir un « produit à succès unique » (*ibid.*), afin de réaliser des économies d'échelle en permettant des déclinaisons autour du socle commun que représente la partie standard du besoin. Cependant, d'autres compétences, portées par d'autres acteurs, assurent les services d'adaptation, de personnalisation et de maintien de la solution en condition opérationnelle (maintenance, suivi de *bug*, adaptation aux évolutions technologiques, etc.).

Or le système de protection du logiciel, comme d'autres systèmes forts de protection intellectuelle (Farrell, 1989), réduit drastiquement les possibilités d'amélioration d'un produit au bénéfice de la seule firme détentrice des droits de propriété. La dynamique d'évolution d'un logiciel à travers ses versions successives reste donc entièrement dépendante des capacités et de la volonté de cette firme, interdisant toute intervention sur le produit, de la part de l'utilisateur ou d'un autre développeur, qu'il s'agisse d'en rectifier les erreurs ou d'en enrichir les fonctionnalités... De même, il sera toujours plus facile pour la firme contrôlant le logiciel de base (un système d'exploitation, par exemple) de proposer des extensions, des logiciels complémentaires, et d'empêcher, par des changements d'interface, des concurrents d'entrer ou de se maintenir sur le marché (en illustration, voir le cas de la concurrence sur les navigateurs *web* entre Microsoft et Netscape, étudié par Gilbert et Katz, 2001). C'est la dynamique concurrentielle de l'industrie du logiciel en général, et de chaque écosystème construit autour d'un logiciel ou d'une plateforme (comme Windows), qui est affectée de par les effets de barrière à l'entrée qui en résultent, et qui sont essentiellement des barrières sur la dynamique d'évolution de la solution technologique et de son écosystème (Bresnahan et Greenstein, 1999 ; Viseur, 2013, sur le cas Mozilla).

C'est là que le modèle de logiciel libre présente des avantages, notamment pour les utilisateurs.

Avantages et inconvénients du logiciel libre pour les utilisateurs

Le cœur de l'incitation à participer à un projet libre est le principe de l'utilisateur-innovateur (Lakhani et von Hippel, 2003 ; von Hippel et von Krogh, 2003) : celui-ci, parce qu'il est le premier bénéficiaire direct de l'innovation, est incité à la produire tout en pouvant s'attendre à des retours d'information sur sa proposition, voire même à des innovations complémentaires et cumulatives. En outre, ce faisant, il bénéficie également d'effets d'apprentissage individuel et renforce, par cette expérience, son propre niveau de compétence (Foray, Thoron et Zimmermann, 2007). Pour un utilisateur doté de compétences informatiques avancées, la disponibilité du code source et la possibilité de réaliser par lui-même ces améliorations est souvent indéniablement moins coûteux que d'attendre que l'éditeur réalise les adaptations dont il a besoin.

En connectant un grand nombre d'individus autour d'un projet commun et sans aucune fermeture, le logiciel libre est ainsi un moyen de tirer parti d'un fantastique potentiel de compétences distribuées.

L'utilisateur-innovateur n'est pas forcément un individu. Il peut aussi s'agir d'une entreprise qui fait développer un logiciel pour ses besoins propres et demande une licence libre pour ne pas être dépendante de son fournisseur (voir l'étude de la stratégie « libre » du département de la Défense des États-Unis par Le Texier et Versailles, 2009), ou qui recherche des collaborations pour partager les coûts de développement (par exemple Google avec Tensorflow[1], Facebook avec Pytorch[2], qui sont les deux solutions *leaders* d'apprentissage machine). Une telle option pouvant, en pratique, donner lieu à l'abandon d'un logiciel précédemment utilisé qui devient trop coûteux à maintenir par rapport à son avantage stratégique (Van der Linden *et al*., 2009), comme le navigateur *web* de Netscape (Ãgerfalk et Fitzgerald, 2007 ; Viseur, 2013). Enfin, plus récemment, on a vu émerger des projets libres planifiés, généralement menés par un consortium d'entreprises, comme OpenStack (Teixeira *et al.*, 2016), afin de créer une norme industrielle, ou de mutualisation par la demande (Elie, 2009), quand les solutions existantes ne sont pas satisfaisantes. Ces différentes approches correspondent bien aux différentes stratégies d'innovation ouverte telles que décrites par West et Gallagher (2006).

Si la production coopérative a toujours existé dans le domaine logiciel, le logiciel libre, comme forme organisée et déclarée, née dès le début des années 1980, a vu son succès remarquablement favorisé par le développement d'Internet, car la plupart des briques de base d'Internet étaient des logiciels libres (Jullien et Zimmermann, 2002), et que le réseau a permis de promouvoir la diffusion et l'adoption de ces logiciels par nombre d'utilisateurs et notamment d'entreprises. S'est ainsi construite une formidable organisation de production, capable de rivaliser avec le mode de production propriétaire.

Les études analysant les qualités (et les défauts) des logiciels libres reprennent les éléments du TCO (Money *et al*., 2012) : sécurité, facilité d'évolution, maintenabilité, testabilité, compréhensibilité, interopérabilité, en plus du prix. D'après ces auteurs, le logiciel libre serait plus efficace du fait de l'ouverture du code, qui facilite la prise en compte des retours utilisateurs (correction de *bugs*, par exemple). Franke et von Hippel (2003) montrent ainsi comment la structuration d'un projet libre (à l'exemple du serveur *web*-Http Apache) repose sur une « boîte à outils d'innovation » permettant à des utilisateurs ayant des compétences variées d'adapter le produit à leurs besoins. Le TCO des logiciels libres serait plus faible, du fait de la disparition des frais de licence mais aussi d'un meilleur respect des standards, qui facilite l'interopérabilité et donc l'intégration dans le système d'information (Almeida *et al.*, 2011). L'étude de Morgan et Finnegan (2014) sur des entreprises adoptant des logiciels libres montre qu'elles ont conscience de ces avantages. On retrouve les mêmes résultats dans l'enquête auprès des DSI des grands groupes français[3] réalisée par le groupe « logiciel libre » du pôle de compétitivité Systematic et du Syntec (association des directeurs des systèmes d'information des grands groupes). Mais les entreprises n'anticipent pas toujours les coûts d'intégration de logiciels libres, qui ne sont évidemment pas rendus nuls (Morgan et Finnegan, 2014) : il s'agit toujours d'intégrer une nouvelle technologique dans un système d'information, donc avec d'autres technologies, et en s'adaptant ou en faisant évoluer des usages individuels et organisationnels. Franke et von Hippel (2003) soulignent aussi que l'implication des utilisateurs est coûteuse et réservée aux utilisateurs les plus compétents, ce qui peut nuire à la prise en compte des utilisateurs non développeurs.

Comme tout système de production et de protection intellectuelles, il ne s'agit pas d'une solution parfaite, mais d'un compromis, porté par certains acteurs (les utilisateurs-développeurs), afin de mieux servir leurs propres intérêts (la gestion à long terme des évolutions technologiques et fonctionnelles d'une solution). La licence du projet est un

[1] https://www.tensorflow.org/

[2] https://pytorch.org/

[3] https://cnll.fr/media/2019_CNLL-Syntec-Systematic-Open-Source-Study.pdf

élément parmi d'autres de régulation de ce compromis. Par exemple, le choix des licences de la Mozilla Foundation se veut un outil pour préserver un équilibre subtil entre les intérêts des développeurs (protection contre le risque d'appropriation), des entreprises (préservation du caractère propriétaire des extensions) et des partenaires (souci de réutilisation) (Viseur, 2013).

Conclusion

Alors que la plupart des modèles classiques de propriété intellectuelle sont basés sur un flux de revenus venant de la monétisation de l'accès à une technologie (le « logiciel stock »), l'idée du logiciel libre est d'utiliser le droit d'auteur pour organiser, *via* des licences spécifiques, la gestion de l'évolution de la technologie et de son interopérabilité (le « logiciel flux »), qui ont autant de valeur pour l'utilisateur que le stock. Cette gestion est garantie par une organisation qui contrôle et fait évoluer les standards d'échange de façon publique, en phase avec l'évolution du logiciel lui-même. Cette double caractéristique d'ouverture et de contrôle permet de construire des normes de définition (David, 1987), référentiel commun de la communauté des développeurs, dont le respect est garant de l'intérêt général du collectif (Benezech, 1995). C'est aussi la gestion de cette évolution et la capacité à l'orienter qui sont à la base des incitations à participer à la production d'un logiciel libre, mais qui donnent lieu aussi à des entreprises de proposer des services *open source* d'installation, d'adaptation, de maintenance des logiciels à des utilisateurs moins experts que les participants aux projets (Jullien et Viseur, 2021).

Cela ne va pas sans difficulté dans l'organisation d'un projet. Ãgerfalk et Fitzgerald (2007) soulignent que construire un projet libre viable est coûteux, même pour les utilisateurs les plus compétents. Il faut notamment arbitrer, trouver le bon équilibre entre une organisation hiérarchique, source d'efficacité productive (Lee *et al*., 2017), et l'ouverture, qui facilite l'intégration de nouveaux membres et donc l'innovation, selon le dilemme classique entre exploration et exploitation d'une solution technologique (March, 1991). Les stratégies des entreprises vendant des prestations commerciales basées sur le logiciel libre ne sont pas toujours alignées non plus avec les stratégies des utilisateurs-développeurs. Cette question de gouvernance des projets est développée dans l'article « Gouvernance d'un projet libre : contrôler un flux d'innovation » (pp. 77-83).

Références bibliographiques

ÃGERFALK P. J. & FITZGERALD B. T. (2007), "Outsourcing to an unknown workforce: Exploring opensourcing as a global sourcing strategy", *MIS Quarterly*, 32(2), pp. 385-409.

ALMEIDA F., OLIVEIRA J. & CRUZ J. (2011), "Open standards and open source: Enabling interoperability", *International Journal of Software Engineering & Applications (IJSEA)*, 2(1), pp. 1-11.

BENEZECH D. (1995), *L'apport du concept de norme technique à l'analyse de la diffusion technologique*, thèse de doctorat, Université de Rennes 1, mention Sciences économiques.

BRESNAHAN T. F. & GREENSTEIN S. (1999)., "Technological competition and the structure of the computer industry", *The Journal of Industrial Economics*, 47(1), pp. 1-40.

CAMPBELL-KELLY M. & GARCIA-SWARTZ D. D. (2015), *From mainframes to smartphones: A history of the international computer industry,* Harvard University Press.

CUSUMANO M. (2004), *The Business of software: What every manager, programmer, and entrepreneur must know to thrive and survive in good times and bad,* New York, Free Press.

DAVID P. A. (1987), "Some new standards for the economics of standardization in the information ages", *in* DASGUPTA P. & STONEMAN (éd.), *Technology Policy and Economic Performance*, Cambridge, Mass, Cambridge UP, pp. 206-239.

ÉLIE F. (2009), *Économie du logiciel libre*, Éditions Eyrolles.

FARRELL J. (1989), "Standardization and intellectual property", *Jurimetrics Journal*, 30, p. 35.

FORAY D., THORON S. & ZIMMERMANN J.-B. (2007), "Open software: Knowledge openness and cooperation in cyberspace", *in* BROUSSEAU É. & CURIEN N. (éd.), *Internet and Digital Economics; Principles, Methods and Applications*, Cambridge University Press.

FRANKE N. & VON HIPPEL E. (2003), "Satisfying heterogeneous user needs via innovation toolkits: The case of Apache security software", *Research policy*, 32(7), pp. 1199-1215.

GILBERT R. J. & KATZ M. L. (2001), "An economist's guide to US v. Microsoft", *Journal of Economic perspectives*, 15(2), pp. 25-44.

HORN F. (2004), *L'économie des logiciels,* La Découverte.

JULLIEN N. & VISEUR R. (2021), "Les stratégies open-sources selon le paradigme des modèles économiques", *Systèmes d'information et management*, 26(3), pp. 67-103.

JULLIEN N. & ZIMMERMANN,J.-B. (2002), « Le logiciel libre : une nouvelle approche de la propriété intellectuelle ? », *Revue d'économie industrielle*, 99, pp. 159-178.

JULLIEN N. & ZIMMERMANN J.-B. (2006), "New approaches to intellectual property: From open software to knowledge based industrial activities", *in* LABORY S. & BIANCHI P. (éd.), *International Handbook on Industrial Policy*, Edward Elgar (EE), pp. 243-264.

LAKHANI K. & VON HIPPEL E. (2003), "How open source software works: Free user to user assistance", *Research Policy*, 32, pp. 923-943.

LE TEXIER T. & VERSAILLES D. W. (2009), "Open source software governance serving technological agility: The case of open source software within the DoD", *International Journal of Open Source Software and Processes (IJOSSP)*, 1(2), pp. 14-27.

LEE S., BAEK H. & JAHNG J. (2017), "Governance strategies for open collaboration: Focusing on resource allocation in open source software development organizations", *International Journal of Information Management*, 37(5), pp. 431-437.

MARCH J. G. (1991), "Exploration and exploitation in organizational learning", *Organization Science*, 2(1), pp. 71-87.

MONEY L. P., PRASEETHA S. & MOHANKUMAR D. (2012), "Open source software: quality benefits, evaluation criteria and adoption methodologies", *Journal of Computations & Modelling*, 2(3), pp. 1-16.

MORGAN L. & FINNEGAN P. (2014), "Beyond free software: An exploration of the business value of strategic open source", *The Journal of Strategic Information Systems*, 23(3), pp. 226-238.

OTA (1992), "Finding a balance; computer software, intellectual property and the challenge for technological change", rapport technique OTA-TCT-527, the Office of Technology Assessment of the United States Congress, Washington, D.C.

SHAIKH M. & CORNFORD T. (2011), "Total cost of ownership of open source software: A report for the UK Cabinet Office supported by OpenForum Europe".

TEECE D., PISANO G. & SHUEN A. (1997), "Dynamic capabilities and strategic management", *Strategic Management Journal*, 18(7), pp. 509-533.

TEIXEIRA J., MIAN S. & HYTTI U. (2016), "Cooperation among competitors in the open source arena: The case of openstack", *Proceedings* of the International Conference on Information Systems, Dublin, Ireland.

VAN DER LINDEN F., LUNDELL B. & MARTTIIN P. (2009), "Commodification of industrial software: A case for open source", *IEEE software*, 26(4), pp. 77-83.

VISEUR R. (2013), "Identifying success factors for the Mozilla project", *Proceedings* of the 9th Open Source Software (OSS) Conference, volume AICT-404: *Quality Verification*, Koper-Capodistria, Slovenia, Springer, Part 1: Full Papers - Innovation and Sustainability, pp. 45-60.

VIVANT M. (1993), « Une épreuve de vérité pour les droits de propriété intellectuelle : le développement de l'informatique », *L'avenir de la propriété intellectuelle*.

VON HIPPEL E. & VON KROGH G. (2003), "Open source software and the 'private-collective' innovation model: Issues for organization science", *Organization Science*, 14(2), pp. 209-223.

WEST J. & GALLAGHER S. (2006), "Challenges of open innovation: The paradox of firm investment in open-source software", *R&D Management*, 36(3), pp. 319-331.

YOST J. R. (2017), *Making IT Work: A History of the Computer Services Industry*, MIT Press.