Le logiciel libre:

une nouvelle approche de la propriété intellectuelle.

Nicolas Jullien - ENST-Bretagne / ICI, Brest et
Jean-Benoît Zimmermann CNRS / IDEP-GREQAM, Marseille.

Introduction

Le débat sur la protection des oeuvre numériques poussé sur le devant de la scène par le développement d'Internet pose le problème de l'irruption des technologies de l'information dans le domaine de la création, qu'elle soit artistique, littéraire ou simplement informationnelle (bases de données) (Vivant, 2001). Ce débat, déjà amorcé par le développement des techniques de reprographie réside dans l'idée qu'une oeuvre puisse être reproduite, sans difficultés techniques majeures et à un coût marginal quasi-nul. Les technologies de numérisation et de compression des données jouent dans ce débat un rôle majeur puisqu'elles fournissent les outils de la codification, de la transmission et de la reproduction.

Le cas du logiciel, qui est par définition une création numérisée rejoint évidemment ce débat, mais en même temps se distingue des produits de contenu qui sont des oeuvres numérisées, alors que la nature des produits logiciels est précisément celle de la logique programmée permettant l'articulation des diverses composantes des architectures matérielles et leur utilisation dans des applications variées. Les questions relatives à la propriété intellectuelle du logiciel et à sa protection se sont posées, dès les années soixante-dix, dès lors que sa commercialisation dissociée, par le fait des législations antitrust, de celle des architectures matérielles en faisait un bien marchand à proprement parler.

Or, tant sur le plan juridique que sur le plan économique, la propriété intellectuelle du logiciel ne trouve de solution satisfaisante dans aucun des cadres existants. La difficulté de construire un cadre spécifique, validé à l'échelle internationale, et le risque de voir ce cadre rendu obsolète par l'évolution rapide de l'activité, ont incité la totalité des législations à classer la protection du logiciel dans la catégorie du droit d'auteur, alors même que plusieurs aspects dans sa nature même, s'avèrent contradictoires avec cette approche de la propriété intellectuelle. Typiquement, le double objectif de la protection qui consiste, d'un côté, à accorder à l'inventeur un monopole temporaire pour l'exploitation de son invention et, de l'autre, à exiger de sa part qu'il dévoile les principes de son invention afin de pouvoir les rendre accessibles à l'ensemble des acteurs technologiques et industriels, n'est pas satisfait. Le recours au système des brevets, amorcé aux Etats-Unis et qui se pose à présent à l'Europe, pose quant à lui d'autres problèmes qui tiennent au risque d'un cloisonnement de la connaissance au sein d'un monde dont la dynamique est fondée sur la cumulativité et le partage du savoir.

Or le modèle du logiciel libre, qui est né dans le même environnement académique des années soixante-dix, tend

dans le même temps à prendre une importance grandissante et à rallier certains parmi ceux-là même qui, comme IBM, se faisaient hier les défenseurs des positions dominantes. Les logiciels libres sont nés, avant même les premières versions de Linux, dans le contexte d'un projet de développement coopératif GNU (GNU Not Unix) pour répondre aux dérives du monde Unix vers la multiplication des versions propriétaires non compatibles (Smets et Faucon, 2000). GNU s'est rapidement doté d'un outil juridique approprié, dit GPL "General Public Licence", qui visait à arrêter les règles du jeu coopératif et empêcher l'appropriation privée des codes. Au *CopyRight* venait s'opposer le *CopyLeft*, dont le principe essentiel était d'imposer, à celui qui l'accepte, de dévoiler le code-source des programmes concernés et de toutes les modifications qui pourraient lui être apportées, mais aussi la libre circulation du code sous la seule restriction de conserver son caractère "ouvert".
Ce modèle, qui produit des logiciels performants et de qualité, constitue une alternative au modèle dominant de production du logiciel, fondé cette fois sur la mutualisation des connaissances. Il induit une dynamique économique fondamentalement différente, autour de la question du développement de logiciel considérée comme une production d'un bien public.

L'objectif de cet article est de mieux comprendre les fondements d'un tel modèle et les conditions de sa viabilité économique. Nous verrons qu'à ce titre il représente ici archétype de ce que l'on pourrait imaginer comme un modèle précurseur dans le cadre d'une économie fondée sur la connaissance. Dans une première partie nous examinerons la nature spécifique du produit logiciel et de l'activité de développement de logiciel. Nous discuterons dans la seconde partie des fondements juridiques de la protection de sa propriété intellectuelle, puis nous examinerons dans la troisième partie les voies de sortir du dilemme entre incitation à l'innovation et diffusion du savoir dans le tissu industriel. Dans la quatrième partie nous examinerons la logique économique du logiciel libre et son support juridique. Dans la cinquième partie, nous essaierons d comprendre les fondements de l'efficacité du modèle et enfin, dans la sixième partie, nous examinerons les conditions de sa viabilité économique, c'est-à-dire de sa pérennité.

1. Le logiciel: un bien et une activité industrielle

Le logiciel doit, sans aucun doute, être considéré aujourd'hui comme un bien marchand, dans la mesure où il peut être produit et commercialisé de manière autonome, indépendamment de la plate-forme matérielle dans laquelle il sera utilisé. Pourtant les nomenclatures courantes, des activités et du commerce, le considèrent encore comme un service et ne reconnaissent pas à sa production le caractère d'activité industrielle. Il faut voir dans cet anachronisme deux explications qui traduisent, d'une part, une transformation relativement récente (moins de trente ans) des conditions d'exercice de l'activité et surtout de commercialisation de ses produits et, d'autre part, la nature particulière de l'activité et surtout des biens qui continuent pour partie d'être réalisés comme des services (développement de logiciels sur mesure).

En premier lieu, donc, l'accès du logiciel au statut de bien marchand a été initié par l'obligation que la législation antitrust, américaine puis européenne, a fait à IBM, à la fin des années soixante, de procéder à une facturation séparée, vis-à-vis des architectures matérielles avec lesquelles il était jusque là lié sur le plan comptable. Dans cette même période, Digital Equipment, en lançant le concept de mini-ordinateur, autonomisait la notion de

système informatique à l'égard des applications que ce dernier était susceptible de porter. Dans le contexte impulsé par cette entreprise et la proximité du cadre universitaire du MIT, se développait alors une activité de production de logiciels d'application, par les nouvelles *software houses*, hors du giron des grands constructeurs d'ordinateurs (Delapierre et Zimmermann, 1994).

En deuxième lieu, le logiciel constitue un bien d'une nature particulière. En termes de coûts, tout d'abord, un bien industriel conventionnel est produit en série et, à côté des coûts de conception et de développement, ce sont avant tout les conditions de coût de sa production en série (importance des coûts fixes, effets d'échelle, production jointe avec d'autre produits, effets d'apprentissage, ...) qui prévalent dans le positionnement concurrentiel des producteurs et en fondent les conditions de tarification. Pour ce qui concerne le logiciel, la quasi-totalité des coûts sont ceux de son développement et, quant à ceux de sa reproduction en série ils sont, en proportion, négligeables et limités aux seuls frais de duplication des supports d'enregistrement, voire nuls dans le cas d'un téléchargement sur Internet. En ce qui concerne la nature de l'objet ensuite, un logiciel, qu'il soit logiciel système ou logiciel d'application, s'apparente d'avantage à une technologie qu'à un produit au sens traditionnel du terme. En tant que logiciel système il est technologie constitutive d'une architecture informatique, en tant que logiciel d'application, il transforme cette architecture en machine dédiée à une ou un ensemble limité de tâches d'utilisation (Gérard-Varet et Zimmermann, 1985). Et le fait est que la commercialisation de ce produit particulier revêt une forme qui découle plutôt de sa nature de technologie, puisque ce n'est pas le produit en lui-même qui est vendu mais un droit d'utilisation non cessible qui est concédé, à travers un contrat de licence implicite à la vente et dont la rémunération est généralement évaluée sur une base forfaitaire. Mais cette technologie que constitue le logiciel se distingue également d'un processus de production dans la mesure où il ne contribue pas directement à la production d'un bien, mais constitue plutôt un processus d'articulation et de coordination de stations logiques dans le cadre d'un objectif de traitement de l'information. De ce fait il est compréhensible qu'il se distingue d'une technologie de production tant sur le plan de la rémunération que sur celui de la protection de la propriété intellectuelle, ainsi que nous le verrons plus loin. Enfin, l'usage d'un produit logiciel ne suppose pas une connaissance approfondie et une maîtrise des processus impliqués et l'accès à son utilisation est obtenue à l'issue d'un apprentissage réduit qui autorise l'utilisateur à tirer profit des fonctionnalités de sa machine sans devoir investir en connaissances informatiques. Plus encore, un logiciel, pour être mis en œuvre, doit être compilé, c'est-à-dire traduit en langage machine (on parle alors de *code-objet*), sous une forme par conséquent très éloignée du langage humain. Bien qu'il ait été généralement conçu dans un langage dit évolué, c'est-à-dire accessible à l'esprit humain, sa formulation dans ce niveau d'expression désigné comme le code-source n'est pas nécessaire à l'utilisateur pour pouvoir bénéficier de l'ensemble de ses fonctionnalités. C'est la raison pour laquelle, le logiciel est commercialisé, dans l'approche propriétaire jusqu'ici dominante, sous la forme de son seul *code-objet*, ne donnant par conséquent pas l'accès aux algorithmes et procédures de programmation. Cette pratique constitue une protection technique de la propriété intellectuelle et le débat n'a jamais été réellement tranché de savoir si la reconstitution d'un code-source par des techniques de *reverse ingeneering* (on parle alors de *décompilation*) constitue ou non une violation de la propriété intellectuelle d'un produit légalement protégé.

En ce qui concerne les conditions de la production des logiciels, Richardson (1997) caractérise l'industrie du logiciel selon quatre traits principaux:

- *coût marginal nul d'usage de la technologie*. En ce sens le coût de développement d'un logiciel n'est pas affecté par l'ampleur de la population des utilisateurs; le coût d'extension marginal de cette population est nul ou réduit à un montant négligeable vis-à-vis du coût de développement.
- *rythme élevé de l'innovation* et par conséquent durée de vie très réduite des produits, ce qui conditionne les modalités de la concurrence, fondée sur la recherche d'obtention rapide de parts de marché, laquelle peut passer par des stratégies de prix agressives, la recherche de rentes monopolistiques fondées sur la protection de la propriété intellectuelle ou la fidélisation de la clientèle à travers des barrières d'incompatibilité (coûts de migration).
- *existence d'effets de réseau*, dans la mesure un logiciel "n'est d'aucune utilité en lui-même, mais seulement quand il est mis en œuvre conjointement avec d'autres produits complémentaires au sein d'un système".
- *rôle des standards* relativement à la mise en œuvre conjointe de produits complémentaires (notamment système d'exploitation et logiciels d'application), ainsi que pour permettre toute articulation et interopérabilité entre programmes, en particulier en ce qui concerne l'échange de données (par exemple entre un logiciel de calcul scientifique et un logiciel graphique ou un tableur, un gestionnaire de base de données et un traitement de texte).

Si l'examen de la nature du produit logiciel montre de toute évidence que celui-ci ne serait en soi éligible ni au droit des brevets, ni à celui du droit d'auteur ou relèverait paradoxalement des deux (Lucas, 1987), l'analyse des caractéristiques de l'industrie conduirait quant à elle à plaider pour un régime de faible protection. Ce sont les arguments qui ont été mis en avant par Joseph Farrell (1992): effets de réseau ou plus récemment par Bessen et Maskin (2000): caractère à la fois séquentiel et complémentaire de l'innovation.[1]

2. Principes et pratiques de la protection du logiciel

En soi, tout aurait dû conduire à la construction d'un outil juridique spécifique de protection de la propriété intellectuelle du logiciel, qui puisse constituer un niveau suffisant d'incitation à l'investissement de la part des éditeurs de logiciels, sans s'inscrire en contradiction avec les caractéristiques de la dynamique de l'innovation dans ce secteur d'activité. Toutefois, la difficulté à élaborer un tel cadre spécifique, validé à l'échelle internationale et la crainte de voir ce cadre rendu obsolète par l'évolution rapide de l'activité (OTA, 1992), ont incité la quasi-totalité des législations concernées à classer la protection du logiciel dans la catégorie du droit d'auteur, alors même que plusieurs aspects, dans sa nature même, s'avèrent contradictoires avec cette approche de la propriété intellectuelle.

Tout d'abord le droit d'auteur ne protège pas les idées, mais une expression particulière de ces idées. Plus encore, comme le souligne, Farrell (1989), ce ne sont pas aux idées utiles que sont reconnues une paternité, mais aux aspects expressifs et arbitraires du travail de développement; "usefull innovations may go unprotected while arbitrary choices of a user interface, for instance, may be held to be protected and may generate large rents if they become de facto market standards". C'est l'adoption seule qui confère à ces aspects arbitraires une valeur sociale qu'ils n'ont pas en eux-mêmes et il n'y a aucune raison économique pour que l'innovateur perçoive une

1 Dans le sens du caractère cumulatif du progrès technique et de l'efficacité de recherches en parallèle fondées sur des approches différentes, respectivement.

quelconque rente sur cet aspect purement arbitraire et non inventif.

Ensuite, du fait qu'un logiciel soit généralement livré sous la forme compilée d'un code-objet, l'expression, protégée par le droit d'auteur, n'est accessible ni aux utilisateurs, ni aux concurrents. Cette occultation apparaît contradictoire avec les objectifs de protection de la propriété intellectuelle qui, dans leur principe, 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). Sur ces questions la jurisprudence américaine est allée dans le sens d'un durcissement plutôt que sur celui de la conciliation, en condamnant pour violation de droit d'auteur des entreprises ayant opéré une décompilation en vue de proposer des produits complémentaires compatibles[2].

Quant au recours au droit des brevets, il avait été jusqu'ici limité en Europe au contexte de procédés industriels brevetables, la protection du logiciel concerné restant alors liée au cadre du procédé de référence. Aux États-Unis au contraire, la jurisprudence à entériné une tendance, de plus en plus marquée depuis les années quatre-vingt-dix, à recourir et obtenir la protection de brevets pour des logiciels, voire pour de simples procédures ou algorithmes. Or, d'une part, l'absence d'un état de l'art dans un domaine où la mutualisation des procédures était jusque là chose courante, y compris dans le secteur commercial, a entraîné un arbitraire dans la délivrance de titres de propriété intellectuelle à l'égard duquel tout programmeur peut être poursuivi pour l'utilisation d'algorithmes ou de procédures considérées jusque là comme faisant partie du domaine public. D'autre part le cloisonnement des savoirs et pratiques qu'un tel cadre de protection entraîne apparaît à l'extrême opposé des préoccupations de préserver la dynamique de cette industrie en lui reconnaissant son caractère de séquentialité et de cumulativité (Scotchmer, 1991).[3]

Cette situation est d'autant plus préoccupante que la complexification des logiciels implique et a été rendue possible par l'utilisation des méthodes modernes de programmation structurée. Les programmes sont construits sur la base d'une combinatoire de modules élémentaires incorporés au sein d'une architecture d'ensemble. Cette approche s'accompagne d'un recours accru à des composants logiciels, portables et réutilisables, et un rapprochement des fondements mathématiques de la programmation. Et l'on est renvoyé de manière de plus en plus systématique à la question de la distinction entre bien public et bien privé en ce qui concerne ces modules et algorithmes. Les législations du droit d'auteur ont exclu du champ de la protection les principes et algorithmes. L'octroi des brevets crée en revanche une barrière à l'utilisation et contredit l'idée de non-exclusion de tels biens, alors même que les pratiques en cours dans l'industrie s'appuyaient sur la cumulativité et l'interactivité.

---

2 C'est par exemple le cas de l'affaire Sega vs Accolade jugée en Avril 1992 à San Francisco, dans laquelle Sega a obtenu que Accolade soit condamné à retirer du marché ses jeux compatibles avec les consoles Sega, du fait de la nécessité qui avait été la sienne d'étudier par reverse engineering une copie temporaire du logiciel d'exploitation de Sega, afin d'en déterminer les conditions de compatibilité. Cette copie a été considérée par la Cour de San Francisco comme une atteinte au droit d'auteur de Sega.

3 Pour un débat plus étendu sur les questions de brevetabilité, nous renvoyons aux autres articles concernés dans ce volume.

3. Deux approches alternatives: normalisation et open-source

Au delà du débat entre droit d'auteur et brevets, on voit clairement que ni l'un ni l'autre ne fournissent de cadre cohérent pour traiter des questions de propriété intellectuelle du logiciel. Ce qui est en cause fondamentalement est le dépassement du dilemme entre incitation à l'investissement des innovateurs et diffusion des connaissances dans le tissu industriel. Dans le contexte spécifique du logiciel, ce dilemme revêt plus particulièrement deux aspects clefs qui conditionnent la dynamique industrielle: l'évolution incrémentale des produits et le problème des interfaces entre produits complémentaires.

Un système fort de protection, qu'il soit fondé sur le droit d'auteur ou sur les brevets, génère un verrouillage des 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... Plus généralement, c'est la dynamique concurrentielle de l'industrie du logiciel qui est affectée de par les effets de barrières à l'entrée qui en résultent. “Entry, competition and innovation may be easier if a competitor needs only to produce a single better component, which can then hook up the market range of complementary components, than if each innovator must develop an entire "system".” (Farrell, 1989)

Le second aspect est donc celui des spécifications d'interface. L'articulation et l'échange de données entre programmes complémentaires nécessitent d'intégrer de part et d'autre les formats et protocoles concernés. Ces questions qui sont plus généralement désignées sous le vocable d'interopérabilité ont été l'objet d'un débat stimulant lors de la préparation de la Directive Européenne au début des années quatre-vingt-dix. Pour favoriser l'interopérabilité et le développement de produits complémentaires compatibles, le texte européen entend contraindre les éditeurs à communiquer les informations nécessaires à la réalisation d'interfaces et, à défaut, autorise la décompilation pour accéder à ces spécifications. Mais à cette volonté claire se heurte une réalité plus complexe, dans la mesure où une décompilation nécessaire à l'analyse des interfaces met également à jour des informations dont la révélation est considérée, par le législateur, comme de l'ordre de la violation du droit d'auteur.

Deux grandes alternatives à ce dilemme peuvent être mises en avant selon que l'on met l'accent sur l'interopérabilité ou sur la dynamique d'évolution des produits. Ces deux alternatives que sont la normalisation et le libre accès aux codes-sources ne sont évidemment pas exclusives l'une de l'autre.

L'approche de la normalisation a été largement mise en avant dans le cadre du débat sur l'interopérabilité, dans l'idée de substituer des standards publics de type interface à des standards de facto, privés, de type produits (Zimmermann, 1995). Elle suit l'idée que les spécifications d'interface ne sont pas de l'ordre de l'activité inventive mais relèvent d'un certain arbitraire et que leur valeur sociale n'est que le résultat d'un processus collectif d'adoption. Dans ce sens la normalisation est un outil de coordination au sein de l'industrie, dont l'efficacité réside cependant dans la mise en phase des arbitrages individuels des firmes avec les arbitrages collectifs à la base des travaux des comités et organismes de standardisation (Farrell et Saloner, 1988).

L'archétype de cette démarche a été cristallisé dans les principes des *systèmes ouverts* et notamment du projet "Posix" développé sous l'égide de l'ISO et de l'IEEE, à la fin des années quatre-vingt, dont le fondement consistait à fournir des standards non-propriétaires spécifiant comment les composants interagissent à leur interface, tout en préservant à ceux-ci la variété et l'aspect boîte noire (Saloner, 1990). Une telle approche s'inscrit à l'opposé d'une logique de construction et de préservation de positions dominantes. Elle permet de concilier l'élaboration de designs propriétaires (donc la propriété intellectuelle) et la variété, avec la pleine réalisation des externalités de réseau sur lesquelles repose la dimension collective de l'industrie, donc de mettre en phase performance individuelle de l'entreprise et efficience collective de la structure industrielle. Dans cette optique, la normalisation peut devenir un facteur mobilisateur de la création de technologie. Porteuse d'efficacité dynamique, elle s'imbrique dans les activités de R&D et s'appuie sur les processus de recherche coopérative inter-firmes (Foray, 1990).

La seconde alternative, à laquelle nous consacrerons la suite de l'exposé, est celle des logiciels libres. Préoccupée principalement par la dynamique d'évolution de l'offre de produits logiciels, elle est fondamentalement basée sur un principe de mutualisation des connaissances, considérées comme de purs biens publics et sur des implications individuelles fondées sur l'action coopérative et la logique don / contre-don (Dang-Nguyen et Pénard, 1999). Par définition, un logiciel libre est un logiciel dont le code-source est rendu ouvertement disponible et ne peut faire l'objet d'une appropriation collective. Ces principes de liberté d'accès aux codes-sources et le développement fondé sur les principes coopératifs qui en découlent permettent la diffusion, dans la communauté des programmeurs et, par delà, au bénéfice de tous les utilisateurs, des meilleures initiatives et progrès en matière de programmation. Dans le même temps, ils permettent l'apport de chacun des programmeurs individuels à la construction d'ensemble. Le modèle des logiciels libres offre la voie d'un mode de conciliation entre intérêt individuel privé et efficience collective. Son efficacité et sa pertinence ne sont évidemment pas exclusives d'une démarche de standardisation fondée sur la diffusion de standards entendus comme des biens publics. Si la voie ouverte pas les systèmes ouverts a été mise à mal par un récent retour aux cloisonnements des systèmes propriétaires, celle proposée par les logiciels libres pourrait contribuer à un dépassement des limites et contradictions des systèmes de propriété intellectuelle et par là même impulser une profonde reconfiguration de l'industrie du logiciel.

4. La logique économique du libre et son support juridique.

Le logiciel libre est donc avant tout une forme réponse aux problèmes de compatibilité et d'évolutivité que pose la fermeture du code source des logiciels. L'enfermement dans des standards propriétaires a, pour les utilisateurs, un double effet. D'une part il limite leur capacité à combiner les outils logiciels en réponse aux spécificités de leurs besoins. D'autre part il les contraint à dépendre du bon vouloir d'un éditeur de publier une nouvelle version d'un logiciel donné, pour espérer y trouver les améliorations et rectifications souhaitées. Pour un utilisateur doté de compétences informatiques avancées, la disponibilité du code-source et la possibilité de réaliser par lui-même ces améliorations serait souvent indéniablement moins coûteux.

Si la production coopérative a toujours existé dans le domaine logiciel, le logiciel libre, comme forme organisée et déclarée, né dès le début des années quatre-vingt, a vu son succès remarquablement favorisé par le

développement d'Internet. Celui-ci a permis de manière spectaculaire de promouvoir la diffusion et l'adoption de ces logiciels par un nombre sans cesse croissant d'utilisateurs et notamment d'entreprises. Il a aussi permis de donner naissance à une formidable organisation de production, capable de rivaliser avec le mode de production propriétaire. C'est l'opposition du *bazar* et de la *cathédrale* (Raymond, 1998) qui stigmatise la confrontation de deux systèmes fondés sur des logiques profondément divergentes.

Depuis quelques années, les logiciels libres se diffusent dans des entreprises utilisatrices et sont intégrés par d'autres entreprises dans leur offre marchande. Ce succès se fonde indéniablement sur une meilleure prise en compte des besoins des utilisateurs et une meilleure gestion des processus de standardisation.

Les succès les plus visibles des logiciels libres sont relatifs aux applications développées autour d'Internet. Ainsi, le logiciel Sendmail achemine environ 80 % du trafic de courrier électronique (Lerner & Tirole, 2000) et le logiciel serveur Apache fait fonctionner plus de 60% des sites Web mondiaux[4]. Dans le domaine des langages de programmation, les logiciels libres sont très répandus: Ada 95, CAML sont, depuis leur création, des logiciels libres et il existe des compilateurs libres pour C/C++, LISP, COBOL ou Fortran. Le langage qui a le plus de succès pour créer des sites Web dynamiques est PHP, qui est aussi un logiciel libre, comme le sont Python ou Perl, d'autres langages utilisés (entre autre) dans les serveurs Web. Mais aussi et surtout, le système d'exploitation Linux équipe aujourd'hui environ 4 % des machines utilisées dans le monde (à comparer aux 5 % détenus par Apple). Au Japon, les prévisions indiquent un marché de 12 % des serveurs en 2001. C'est actuellement le système d'exploitation qui connaît le plus fort taux de croissance.

Côté offre, un exemple significatif par excellence est celui d'IBM. Après avoir annoncé qu'Apache serait désormais le logiciel serveur de son offre “e-business”, IBM a également déclaré son intention d'investir un montant d'un milliard de dollars dans le développement et la commercialisation de Linux pour l'année 2001. Un tel investissement, de même que ceux annoncés par d'autres firmes du secteur marchand a de quoi surprendre et relève du paradoxe, puisque le fondement du logiciel libre réside dans le caractère inappropriable des codes concernés. Avant de chercher à comprendre la logique de ces investissement privés, il nous faut décrire l'outil p central, garant de ce caractère public inaliénable, la licence GPL, sur laquelle se fonde la spécificité du logiciel libre par rapport à d'autres productions coopératives. La licence GPL ne remet pas en cause la propriété intellectuelle en elle-même, mais en propose une gestion radicalement nouvelle. Le principe fondamental en est que, les codes concernés étant libres, tout programme qui intègre des lignes de code GPL doit aussi être disponible sous licence GPL. Ainsi les auteurs n'abandonnent pas leur paternité intellectuelle, mais la seule rente de monopole qu'autoriserait un régime de copyright. Le logiciel reste de la sorte la propriété de ses créateurs. Ils autorisent quiconque à en faire usage (modifications, améliorations, compléments...) sous la seule condition que toute nouvelle version puisse, elle aussi, circuler librement.

En soi, la licence GPL constitue une adaptation du système de protection intellectuelle, qui vise à corriger les excès du système propriétaire (Clément-Fontaine, 1999). Dans les deux systèmes, libre et propriétaire, la relation utilisateur-producteur repose sur le droit d'auteur qui confère un droit de propriété sur un logiciel à son producteur et lui permet de le licencier à un utilisateur en lui imposant des conditions restrictives d'utilisation. Dans le cas du logiciel propriétaire, un droit d'utilisation est concédé en échange d'une rémunération. Dans le cas du logiciel libre les conditions restrictives ne portent pas sur l'utilisation individuelle, mais sur la diffusion et l'amélioration du logiciel : le licencié doit répercuter en aval les mêmes libertés, que le logiciel ait ou non été modifié. Par voie de conséquence, il n'est plus possible, pour un producteur, de se rémunérer directement par la vente du logiciel : puisque chacun peut le redistribuer, les utilisateurs peuvent concurrencer le producteur et finalement le logiciel ne peut être vendu qu'à un coût proche de son coût marginal.

4 Voir http://www.netcraft.com/survey/ sur la part de marché d'Apache.

On peut dès lors s'interroger sur les motivations qui peuvent conduire des entreprises à vocation commerciale à s'impliquer dans le modèle du libre. Ces questions que nous allons éclaircir dans les deux sections suivantes renvoient également à celle du régime de propriété intellectuelle. En réalité un certain nombre d'entreprises ont pris soin de se démarquer de la licence GPL en proposant leurs propres licences, composés hybrides du copyright et du *copyleft*[5]. Ainsi Netscape en "libérant" son *Communicator* se réserve le droit d'y intégrer des modules propriétaires, ou Sun se réserve l'exclusivité de la certification en compatibilité de tout produit issu de son protocole de communication *Jini*. Pourtant ces approches, souvent contestée, finissent par se révéler plus bâtardes qu'hybrides, cherchant à concilier l'inconciliable plutôt que de faire clairement un choix entre deux approches que tout oppose. Ainsi Sun semble aujourd'hui se résoudre à adopter sans restriction la GPL, notamment pour son système d'exploitation *Solaris* et pour la suite bureautique *Star Office*, qu'elle contrôle.

5. Le mode de production libre, source de produits performants et de qualité.

Une des raisons qui expliquent le succès du logiciel libre est relative à son organisation très structurée. La coordination entre les différents développeurs est le plus souvent assurée par un groupe de personnes, que nous appellerons les “développeurs clefs” et qui supervisent la plus grande partie des développements du logiciel concerné[6]. Les développeurs clefs sont sélectionnés par les autres membres du noyau de développement (c'est le cas d'Apache) ou par le développeur initial du projet (c'est le cas de Linux), mais il y a toujours cooptation à l'instar du milieu scientifique. Ce noyau reste ouvert aux nouvelles idées, aux nouveaux talents : on peut confier à de nouveaux venus la responsabilité de parties nouvelles du logiciel, souvent parce que ce sont eux qui ont commencé à les développer. A la périphérie du noyau de développement, de nombreux utilisateurs, souvent eux-mêmes développeurs participent au processus, en donnant des idées, en proposant des améliorations ou en détectant des erreurs[7]. L'existence de ce groupe élargi est importante pour le succès et l'évolution du logiciel mais il n'a que peu d'impact direct sur le volume de logiciel développé.

La deuxième condition nécessaire au bon fonctionnement d'un projet libre est plus technique. Elle porte sur l'architecture du logiciel, qui doit être construit en modules. Les frontières du logiciel de base sont définies, les interfaces de communication entre ce logiciel de base et les logiciels satellites, les “composants”, sont rendues publiques. Les développeurs sont alors encouragés à produire des “composants” qui s'additionnent au coeur du logiciel pour offrir de nouvelles fonctionnalités, indépendamment les uns des autres. La construction modulaire

5 ainsi désigne-t-on souvent les principes de gestion de la propriété intellectuelle dont la GPL est la traduction juridique.

6 L'étude de Mockus & al. [2000] sur le projet Apache souligne cet aspect : “these results indicate that despite broad overall participation in the project, almost all new fonctionality is implemented and maintained by the core group” (p.268).

7 “Of the top 15 problem reporters only three are also core developers. It shows that the significant role of system tester is reserved almost exclusively to the wide community of Apache users” (Mockus & al. [2000], p. 269).

d'un logiciel permet à des groupes de développeurs de travailler relativement indépendamment. Ainsi les équipes de programmation restent de petite taille, ce qui facilite la coordination et la prise de décisions[8]. Par ailleurs cette organisation modulaire facilite les innovations et l'extension des fonctionnalités des programmes, qui peuvent se faire par des ajouts successifs de modules[9].

Ainsi est-on assuré d'une certaine stabilité de l'évolution du noyau et de l'interopérabilité entre les différents modules. En cela, l'organisation du libre traite le problème de l'interopérabilité de façon originale : elle 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. Elle génère un processus dynamique de standardisation et d'évolution du standard, qui conserve un caractère public pur. Plus largement, 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 de la collectivité (Benezech, 1995). Sur le plan cognitif elle génère, à travers la dynamique d'interaction, une communauté épistémique comprise comme le partage par un groupe d'un objectif commun de création de connaissance et d'un environnement d'interaction permettant à chacun des individus de s'insérer en phase avec cette dynamique collective (Cohendet, Creplet et dupouëy, 2001).

Parce qu'elle permet aux utilisateurs d'étudier le code et de tester les logiciels, mais aussi parce que les règles de développement d'un projet libre sont souvent très contraignantes[10], parce qu'enfin chaque proposition doit être acceptée par le noyau de contrôle, l'organisation du libre produit ainsi des logiciels performants. Leurs qualités technico-économiques (stabilité, respect des standards couplé à une évolutivité des solutions) sont aujourd'hui reconnues[11] et expliquent leur adoption par une population élargie d'utilisateurs, bien au delà de la seule communauté des individus impliqués directement ou indirectement dans des tâches de développement.

Là sont aussi les raisons pour lesquelles plusieurs entreprises du secteur propriétaire ont décidé de basculer dans le monde du libre pour tout ou partie de leurs produits. Ainsi Sun est aujourd'hui le deuxième producteur de logiciel libre, après la FSF[12], mais avant les universités américaines. Par ailleurs de nombreuses entreprises ont vu le jour, souvent à l'initiative d'individus fortement impliqués dans le monde du libre, en vue d'accompagner le développement d'un logiciel donné par une offre de services associés (RedHat ou Mandrake avec Linux, Covalent Technology avec Apache, Helixcode avec Gnome, une interface graphique de Linux, etc.).

8 “In most open source projects” says Zawinski, “there is a small group who do the majority of the work, and the other contributors are definitely at a secondary level, meaning that they don't behave as bottlenecks.” (cité par Jones [2000], Zavinski a travaillé chez Netscape et au projet Mozilla).

9 voir Bowman [1998] pour une présentation de la répartition des responsabilités dans le développement du noyau Linux.

10 Voir http://www.gnu.org/prep/standards.html pour les recommandations techniques à suivre pour programmer des logiciels GNU.

11 Voir Lefèvre [2001] pour une analyse plus technique des raisons de la meilleure efficacité des solutions libres.

12 Sur l'étude des contributions des différents producteurs de logiciels libre, on consultera la remarquable enquête d'Orbiten (http://www.orbiten.org/ofss/01.html) et Jullien [2001] (chapitre 4).

6. La viabilité d'un modèle de propriété intellectuelle fondé sur l'ouverture et la mutualisation

De par son caractère codifié, un bien informationnel comme le logiciel doit-il ou tout simplement peut-il être considéré comme un bien public? En d'autres termes, un modèle de gestion de la propriété intellectuelle comme celui avancé par le logiciel libre est-il en mesure de proposer une alternative pérenne au modèle propriétaire, une alternative viable aussi bien pour les développeurs individuels mobilisés, que pour les utilisateurs et pour les entreprises nées du logiciel libre ou qui ont décidé de s'y rallier?

les développeurs: un problème d'incitations

Lorsque des utilisateurs-développeurs produisent un logiciel pour leurs besoins propres, il est peu coûteux de le publier en libre et peu d'avantages peuvent être espérés d'une non-divulgation. Mieux encore, la publication de leur travail peut générer des effets en retour de la part des autres développeurs : critiques, détection d'erreurs et suggestions permettent de faire évoluer le produit. Lorsque les nouveaux utilisateurs du logiciel sont aussi développeurs, ils peuvent directement s'impliquer dans le développement de nouvelles fonctionnalités, d'où résulte une mutualisation des coûts, en retour de la mutualisation des connaissances. Ici la question d'une éventuelle appropriation privée des fruits du développement ne se pose pas vraiment car il n'y a pas de véritable rivalité entre les utilisateurs producteurs de logiciel à ce stade.

La programmation de systèmes complexes comme les systèmes d'exploitation, les compilateurs ou les grands programmes de calcul, fait appel à une diversité de compétences qu'aucun développeur individuel ne peut prétendre posséder. Si le nombre de développeurs impliqués dans un noyau de développement reste peu élevé, les individus peuvent s'y singulariser par leurs talents et leurs compétences propres et les échanges restent fondés sur une logique de don / contre-don.

Mais le succès du logiciel libre va bien au-delà. En soi, si la production de logiciel libre se limitait à une audience d'utilisateurs-développeurs hors de la sphère marchande, il ne répondrait qu'aux besoins d'une petite partie de la population et ne constituerait pas un phénomène d'intérêt économique. En revanche, comme le montrent Foray & Zimmermann (2001), l'arrivée de nouveaux adopteurs, d'une nouvelle demande, qui tire bénéfice du processus sans y contribuer, risque de décourager la mobilisation des développeurs et donc de nuire à l'efficacité du processus coopératif. On est confronté ici à un problème de défaut d'incitations, dans lequel les effets espérés, en termes de réputation et de compétence, pour le programmeur devront être complétés, à partir d'un certain niveau par des incitations monétaires, en termes de rémunération, quelles soient d'origine privée (entreprises impliquées) ou publique (politique technologique). Derrière ce problème, se pose également la question plus générale des conditions de la migration des simples utilisateurs vers le libre. Nous verrons que ceux-ci ont intérêt à jouer, avec les développeurs, le jeu de la coopération.

les utilisateurs: migrer vers le libre ?

Ce sont par les utilisateurs que le libre est sorti de la sphère universitaire où il a été développé. Parce que ces utilisateurs ont les mêmes besoins de garantie de la pérennité de leurs investissements, de garantie de la compatibilité des logiciels, parce qu'ils sont souvent dans un rapport de dépendance vis-à-vis des producteurs de logiciels en situation dominante, ces utilisateurs peuvent être intéressés par l'utilisation du libre.

Pour les utilisateurs dotés de capacités informatiques avancées, la disponibilité des codes sources leur permet de pourvoir directement à l'adaptation des logiciels libres à leurs besoins ou situations spécifiques. Ces adaptations peuvent aussi le cas échéant comprendre le développement de nouvelles focntionnalités. Dans tous ces cas, ces utilisateurs "sophistiqués" (Gérard-Varet et Zimmermann, 1985) deviennent eux-mêmes développeurs. Ainsi que le montre von Hippel (1988), dans de nombreuses situations, il peut être plus intéressant pour eux de contribuer aux efforts collectifs de développement du libre en publiant les modifications qu'ils y apportent, plutôt que de les garder secrètes.

Pour les simples utilisateurs, la question du ralliement au monde du libre n'est pas directement liée à la disponibilité des codes-sources. En revanche le contexte d'évolutivité et de compatibilité des produits du libre les dégage d'une dépendance vis-à-vis du ou des éditeurs de logiciels, de la bonne volonté desquels ils dépendent, tant sur le plan de l'amélioration des versions disponibles que sur celui des possibilités de combinaison d'outils (interopérabilité). La question centrale pour eux se pose en termes de coût de migration par le fait que le passage d'une catégorie de standards à une autre pose le problème de la migration des produits d'application existants (données, textes, statistiques, objets formatés, programmes ...), mais aussi de la remise en cause des externalités de réseau issues des possibilités d'échange entre utilisateurs partageant un même standard et enfin des efforts d'apprentissage consécutifs à l'adoption d'un nouveau standard.

Ces problèmes de migration recouvrent en réalité deux aspects complémentaires. En premier lieu ils comprennent un aspect de dynamique, dans la mesure où des effets de rendements croissants d'adoption (Arthur, 1989) ont pour conséquence une diminution radicale des coûts de migration, au fur et à mesure du ralliement des utilisateurs. En revanche, pour les premiers utilisateurs qui décident de migrer vers le libre, il faut compter que ces coûts de migration soient compensés soit par des avantages en termes de fonctionnalités d'utilisation, soit par une solide capacité d'anticipation et de croyance sur le devenir du libre. Evidemment, en deuxième lieu, cette difficulté peut se voir très fortement estompée par la possibilité de pouvoir passer d'un monde de compatibilité à un autre, à un coût faible et sans entraîner de trop fortes distorsions. Cette possibilité est contingente de l'existence de "technologies passerelles" (*gateway technologies*) dont l'objet est de permettre de faire passer un objet informationnel formaté en conformité avec un monde de compatibilité A vers un monde de compatibilité B et réciproquement. La possibilité de pouvoir émuler un environnement Windows comme "bureau" ouvert sous Linux et dans le contexte d'une gestion multitâches, constitue bien entendu une avancée importante dans ce sens. Mais les passerelles sont encore limitées et de leur existence dépendra la possibilité de constituer le monde du libre en alternative viable à celui d'un monde de production de logiciels fondé sur l'appropriation privée et le

monopole d'exploitation des connaissances développées.

Les entreprises: opposition entre deux modèles.

Il existe d'abord des motivations indirectes qui peuvent inciter certaines entreprises à contribuer au développement de logiciels libres. Ces motivations sont liées aux effets concurrentiels des stratégies de contrôle des standards. D'un côté, on peut viser d'affaiblir un concurrent par l'intermédiaire d'une subvention à un standard antagoniste à celui sur lequel s'appuient son offre de produits. A l'inverse, les entreprises qui dépendent d'un standard pour réaliser leur activité (comme Sun avec Apache) ont tout intérêt à ce que ce standard soit le plus ouvert possible, afin ne pas dépendre de la stratégie de l'entreprise qui le fournit. Elles rejoignent ainsi la logique d'autres entreprises qui ouvrent très volontairement les standards dont elles sont porteuses afin d'en favoriser l'adoption, ainsi que l'avait fait Adobe concernant son standard d'impression Postscipt. De telles motivations peuvent expliquer le ralliement d'entreprises au monde du libre. Mais comme le font remarquer Genthon & Phan (1999) ou Lerner & Tirole (2000), le risque est grand que la participation de ces entreprises soit limitée au temps où elles trouvent un intérêt stratégique à affaiblir leurs concurrents.

Plus directement et plus fondamentalement, les motivations à participer au monde du libre, peuvent recouvrir un tout autre ordre de préoccupations qui induit une reconsidération du fondement de l'activité de ces entreprises. Cette reconsidération est relative aux complémentarités produits-services dans la relation entre utilisateur et éditeur d'outils logiciels. Les modalités de mutualisation des connaissance que garantit la licence GPL transforment en profondeur les relations entre utilisateurs et éditeurs et entre éditeurs eux-mêmes. Tour d'abord les relations entre éditeurs et utilisateurs se recentrent sur une meilleure adaptation du produits aux spécifications d'applications, par l'adaptation du produit aux conditions de l'utilisation, le développement de nouvelles fonctionnalités, l'interopérabilité avec d'autres logiciels. En cela cette relation est source d'innovation en retour sur le produit et la qualité de la réponse à la demande de l'utilisateur est le fruit non seulement des performances intrinsèques du produit mais aussi et surtout de la prestation de services associés autour de ce produit, prestation de services qui, au delà de l'amélioration et de l'adaptation du produit, concernent aussi des aspects de formation, manuels d'aide, aide en ligne, maintenance ... Cette prestation est d'autant plus génératrice de chiffre d'affaire que le produit logiciel de base est facturé proche de son coût marginal et que les services associés amplifient de manière significative la satisfaction du client.

La licence GPL constitue un garant fondamental de l'engagement de ces entreprises. N'étant plus en situation de monopole sur la production et la distribution d'un logiciel donné, elles porteront leurs efforts sur la relation de service, laquelle devient le seul facteur de différenciation de l'offre, susceptible de fonder la fidélité de la clientèle[13]. L'organisation du logiciel libre, fondée sur la GPL, est ainsi un système qui permet de résoudre une partie du hasard moral qui caractérise l'industrie du service et particulièrement du service informatique (De Bandt, 1998).

Entre les entreprises ensuite, se créent alors des relations de coopération, médiatisées par l'organisation de

13 à l'extrême opposé des stratégies de fidélisation fondées sur l'enfermement par l'incompatibilité et les coûts de migration.

développement du libre, chacune d'entre elles mettant en avant un actif spécifique issu de la connaissance des besoins d'utilisation de sa propre clientèle. Ces actifs spécifiques se combinent aux compétences des autres acteurs du développement pour générer l'évolution des produits dans un contexte où l'ouverture des interfaces est garantie par la GPL

Conclusion.

Considérer ou non les connaissances comme un bien public peut difficilement être la conséquence de considérations relatives à leur nature de connaissances pures ou de connaissances appliquées, catégories entre lesquelles il serait bien illusoire de vouloir déterminer une frontière[14]. Certaines connaissances tacites ne sont certes transférables qu'à l'issue d'un processus dans lequel les deux parties doivent s'impliquer, mais on trouve ce type de problème aussi bien au plus amont scientifique dans l'habileté à construire la preuve d'un théorème que dans le monde le plus appliqué d'une opération d'usinage industriel ou de lissage d'un enduit dans le bâtiment. Dès lors qu'une connaissance est susceptible de circuler librement sans se heurter à des problèmes techniques de codage ou de communication, c'est-à-dire peut se ramener à une information, il n'y a pas de raison de ne pas considérer qu'elle constitue un bien public. Seules des considérations d'éthique (le génome humain et la brevetabilité du vivant) et des considérations économiques (incitation à l'innovation) peuvent motiver une décision de protéger ou de ne pas protéger la propriété intellectuelle d'une connaissance.

Ce qui est intéressant dans le cas du logiciel libre est que ce mode de gestion ouvert de la propriété intellectuelle soit à l'origine d'une efficacité indéniable en termes de qualité et de performances des produits. En soit les approches fondées sur la mutualisation des connaissances dans un monde marchand ne sont pas nouvelles et on les retrouve dans la notion de *collective invention* telle que analysée par Allen (1983) à propos de la métallurgie dans le Landcashire. Plus loin encore Foray et Hilaire-Perez (2000) ont décrit la manière dont les Canuts lyonnais avaient mis en place un système circulation des savoirs fondé sur le financement collectif d'une rémunération individuelle des inventeurs. En ce qui concerne le logiciel libre l'enjeu se situe bien entendu dans la possibilité d'atteindre une viabilité économique de long terme. Nous en avons examiné les conditions et il est certain que leur réalisation dépend notamment d'un certain nombre de dimensions institutionnelles telles que les actions de politique technologique, mais aussi les pratiques de protection de la propriété intellectuelle.

Si le problème n'est pas nouveau, il n'en demeure pas moins que, si tant est que nous entrons dans l'ère d'une économie fondée sur la connaissance, cette question de la mutualisation des connaissances, dans un contexte marchand, a toute les chances de se voir posée dans un nombre croissant de domaines et d'activités. Ainsi en est-il dès aujourd'hui dans le cadre du développement thérapeutique de certaine affection comme le cancer, mais demain plus généralement dans le monde immense qui s'ouvre à la génomique. Ici le logiciel libre fait figure de modèle précurseur, qui pose très nettement le problème de l'alternative entre deux modes ou deux mondes de production (Horn, 2000), fondés sur des conceptions radicalement opposées de la propriété intellectuelle. Il y

14 faut-il considérer que la recherche de nouveaux nombres premiers, à seule fin de nourrir des algorithmes de cryptographie, est productrice de connaissances pures ou de connaissances appliquées?

aura certainement lieu de s'appuyer sur cet exemple pour tenter de cerner ce qu'il risque d'en advenir dans d'autres domaines confrontés demain à un problème similaire.

Bibliographie:

Allen R. (1983), "Collective invention", *Journal of Economic Behavior and Organization*, 4, 1-24

Arthur B. (1989), "Competing technologies, increasing returns and lock-in by historical events", The Economic Journal, March.

Benezech D. (1995), *L'apport du concept de norme technique à l'analyse de la diffusion technologique*, Thèse de doctorat de l'université de Rennes 1, mension Sciences Économiques.

Bessen J. et Maskin K. (2000), "Sequential Innovation, Patents and Imitation", MIT, Department of Economics, *Working Paper N°00-01*, January.

Bowman I. (1998), «Conceptual Architecture of the Linux Kernel», http://plg.uwaterloo.ca/\textasciitilde{}itbowman/CS746G/a1/

Clément-Fontaine M. (1999), "La Licence Publique Générale GNU", Mémoire de DEA "Droit des créations immatérielles", Université de Montpellier I, http://crao.net/gpl/ .

Cohendet P., Creplet F. et Dupouët O. (2001), "Organisational innovation, communities of practice and epistemic communities: the case of Linux", in Kirman A. and Zimmermann J.B. (Eds.), *Economics with Heterogenous Interacting Agents*, Springer.

Dang-Nguyen G., Pénard T.(1999), "Don et coopération dans Internet: une nouvelle organisation économique?", in Desbois, Jullien, Pénard, Poulain-Maubant, Vétois et Zimmermann (eds.), "Logiciels Libres: de l'utopie au marché", Numéro Spécial, Terminal, n°80-81, automne-hiver, pp.95-116.

David P.A. (1987) "Some new standards for the economics of standardization in the information age", in Dasgupta & Stoneman eds., *Technology policy and economic performance*, Cambridge UP, Cambridge, Mass., pp. 206-239.

De Bandt J. (1998), "Les marchés de services informationnels~: quelles garanties pour le client, consommateur ou partenaire ?", revue d'économie industrielle, n°86, 4e trimestre, pp. 61-84.

Delapierre M., Zimmermann J.B. (1994), "La globalisation, une remise en perspective des structures techniques de l'industrie", Terminal, n°66, Hiver, pp. 89-109.

Farell J. (1989), "Standardization and Intellectual Property", *Jurimetrics Journal*, Fall.

Farrell J. (1992), "Some Arguments for Weaker Intellectual Property Protection in Network Industries", Dept of Economics, University of California, Berkeley, For presentation at TPRC, September.

Farrell J. and Saloner G. (1988), "Coordination through commitees and markets", Rand Journal of Economics, vol.19, N°2, Summer, 235-252

Foray D. (1990), "Exploitation des externalités de réseau versus évolution des normes~: les formes d'organisation face au dilemme de l'efficacité dans le domaine des technologies de réseau", Revue d'économie industrielle, n°51, pp. 113-140.

Foray D. et Hilaire-Perez L. (2000), "The economics of open technology: collective organization and individual claims in the "fabrique lyonnaise" during the old regime", *Conference in honor of Paul David*, Turin, Mai.

Foray D. et Zimmermann J.B. (2001), "L'économie du logiciel libre: organisation coopérative et incitation à l'innovation", *Revue Economique*, à paraître

Gérard-Varet L.A., Zimmermann J.B. (1985), "Concept de produit informatique et comportement des agents de l'industrie", contribution au colloque *Structures économiques et économétrie* - Lyon, 23 et 24 Mai.

Hipple (von) E. (1988), *The Sources of Innovation*, Oxford University Press, New York.

Horn F. (2000), *L'économie du logiciel. Tome 1 : De l'économie de l'informatique à l'économie du logiciel. Tome 2 : De l'économie du logiciel à la socio-économie des "mondes de production" des logiciels*, Thèse de doctorat d'économie industrielle, Université de Lille I, 570 p.

Jones P. (2000), «Brooks' Law and open source: the more the merrier? Does the open source development method defy the adage about cooks in the kitchen?, mai, http://www-106.ibm.com/developerworks/library/merrier.html.

Jullien N. (2001), *Impact du logiciel libre sur l'industrie informatique*, Thèse de doctorat de sciences économiques, Université de Bretagne Occidentale, 325 p.

Lefèvre A., “ pourquoi les logiciels libres sont meilleurs? ”, http://solutions.journaldunet.com/0101/010111_decrypt_oss.shtml. (TITRE À VÉRIFIER)

Lerner J. et Tirole J. (2000), "The simple Economics of Open Source", Harvard Business School et Institut d'Economie Industrielle *Mimeo*

Lucas A. (1987), “Le Droit de l'informatique”, Thémis, PUF, Paris.

OTA (1992), “Finding a Balance; Computer Software, Intellectual Property and the Challenge for Technological Change”, Rapport Office of Technology Assessment - OTA-TCT-527, Washington DC.

Mocus

Raymond E.S. (1998), *La cathedrale et le bazar*, traduit par Blondeel S., http://www.lifl.fr/~blondeel/traduc/Cathedral-bazaar/Main_file.html .

Richardson G.B. (1997), “Economic analysis, public policy and the software industry”, DRUID Working Paper, n°97-4, avril, in The Economics of Imperfect Knowledge - Collected papers of G.B.Richardson, Edward Elgar.

Saloner G. (1990), "Economic issues in computer interface standardization", Economics of Innovation and new Technologies, vol.1, pp.135-156.

Scotchmer S. (1991), "Standing on the shoulder of giants: cumulative reserach and the patents law", *Journal of Economic Perspectives*, vol.5, n°1, hiver, pp.29-41

Smets-Solanes J.P. et Faucon B. (1999), Logiciels libres - Liberté, égalité, business, Edispher, Paris

Stallman R. M. (1998), ``The GNU Project'', ``Open Sources'', O'Reilly.

Vivant M.(1993), "Une épreuve de vérité pour les droits de propriété intellectuelle: le développement de l'informatique", in *L'avenir de la propriété intellectuelle*, Librairires Techniques, Collection le Droit des Affaires, Paris.

Vivant M. (2001), "Propriété intellectuelle et nouvelles technologies", Géo-économie, n°17, printemps.

Zimmermann J.B. (1995), "L'industrie du logiciel, de la protection à la normalisation", in Baslé M., Dufourt D., Héraud J.A. et Perrin J., "Changement institutionnel et changement technologique", CNRS Editions, Paris.