Quelques éléments d'économie du logiciel libre *

Jacques Crémer †
Alexandre Gaudeul ‡

30 mars 2004

1 Introduction

Le terme 'logiciel libre' désigne à la fois un bien – le logiciel – indispensable au fonctionnement des sociétés modernes informatisées, et un mode de production et de distribution – libre. Ce mode de production et de distribution est suffisamment original pour poser des problèmes nouveaux à la fois d'analyse et de politique économique. Le but de cet article est de présenter une introduction à quelques unes des analyses économiques du logiciel libre et de ses conséquences. Il ne s'agit pas de faire un tour d'horizon complet du sujet, mais de discuter quelques points qui nous semblent particulièrement intéressants. Il ne faut donc pas chercher ici une discussion complète du sujet, mais plutôt un point de vue subjectif consistant à mettre en valeur quelques aspects de la littérature.

Une première partie de ce papier définira le concept de logiciel libre : le logiciel libre se définit par l'ensemble des règles régissant son utilisation et son développement. Ces règles sont exprimées dans la licence à laquelle son utilisation doit se conformer. Une deuxième partie s'attache à étudier les raisons qui poussent à développer et utiliser des logiciels libres, et montre comment programmeurs, entreprises et usagers forment un réseau aux intérêts parfois contradictoires dont les interactions seront décrites. La troisième partie explique comment un phénomène longtemps

*Cet article a été préparé pour l'Ecole thématique CNRS "Technologies de l'Information et de la Communication et structuration des collectifs" organisée par le GDR TICS, Carry le Rouet, 7–12 septembre 2003. Nous remercions Jean Tirole pour ses commentaires sur une version préliminaire, et le programme interdisciplinaire "Société de l'Information" du CNRS pour son soutien.

†CNRS, GREMAQ & IDEI, Université de Toulouse, jacques@cremeronline.com
‡Universités de Toulouse & Southampton, alexandre.gaudeul@univ-tlse1.fr

marginal est devenu un problème de politique économique, et exposera un certain nombre des décisions que la puissance publique doit prendre.

2 Définition et problèmes

Le logiciel libre est ouvert et gratuit.[1] Il est ouvert parce que son code – l'ensemble des instructions qui définissent son comportement – est disponible dans une forme qui permet d'en comprendre le fonctionnement et d'y apporter des changements. En cela, il s'oppose à la plupart des logiciels propriétaires[2] qui ne sont livrés que sous forme compilée, compréhensible pour un ordinateur mais pas par un humain. Ce caractère ouvert ne recouvre pourtant pas toute sa définition – il existe des logiciels commerciaux ouverts mais dont le code ne peut être modifié sous peine de sanctions. Le logiciel libre ne se contente pas d'ouvrir le code, il autorise aussi les modifications. Cette possibilité est ouverte à tous sans discrimination. Les promoteurs de logiciel libre insistent sur ce double aspect de la liberté, et sur le fait que la gratuité n'est pas une condition suffisante pour qu'un logiciel soit considéré comme logiciel libre. Richard Stallmann, qui popularisa le concept, aime à le rappeler à travers la formule *"Free as in free speech, not as in free beer"*.

Avant d'introduire et de discuter une définition du logiciel libre dans la section 2.1, il peut être nécessaire de replacer briévement le logiciel libre dans son contexte historique et institutionnel: Richard Stallmann créa la Free Software Foundation ('FSF', `http://www.fsf.org`) au milieu des années 80 afin de contrer ce qu'il considérait comme une évolution néfaste de l'industrie du logiciel. Alors qu'il jusque là était possible d'accéder au code source de la plupart des logiciels, l'expansion du marché informatique et sa standardisation ouvraient de nouvelles perspectives de profit pour les développeurs, l'accès à un marché de masse les poussant à protéger leur travail en le livrant sous forme compilée. Cette protection n'était pas nécessaire auparavant, chaque logiciel étant réalisé sur commande pour une application ou une machine précise. Le but de Stallmann était de produire un système d'exploitation libre, de façon à garantir à tous les citoyens-usagers le contrôle de leur outil de travail.

À ces motivations idéologiques sont venues se rajouter des motivations plus pratiques, dont l'Open Source Initiative ('OSI', `http://www.opensource.org`) s'est faite la représentante. L'OSI souligne les avantages en terme d'efficacité du mode de développement libre. L'opposition FSF/OSI se traduit schématiquement dans l'opposition entre deux types de licenses

1. Comme nous l'expliquons plus bas, on peut vendre du logiciel libre. Par contre, on n'a pas le droit d'en empêcher une distribution gratuite.

2. On définira ici le logiciel propriétaire comme tout logiciel qui n'est ni libre, ni de domaine public. Cela correspond à l'usage dans la communauté libre. Notons tout de même que le terme 'propriétaire' désigne aussi en français un logiciel développé à l'intérieur d'une entreprise pour ses besoins particuliers.

libres, la General Public License (‘GPL’) et la Berkeley Software Distribution (‘BSD’), mais nous reviendrons plus tard sur ce point. L’OSI a contribué à définir de façon précise ce qu’est un logiciel libre en termes pratiques, et c’est cette définition qui sera discutée dans la section 2.1.

La plupart des projets libres sont listés sur SourceForge (`http://sourceforge.net`) et/ou Freshmeat (`http://freshmeat.net`). Ces sites sont des lieux de rencontre et de travail pour les développeurs libres. SourceForge liste plus de 60,000 projets sur son site, pour la plupart destinés aux plates-formes Unix et Linux (par opposition à Windows).

Le nombre moyen de développeurs pour un projet est de 4, la médiane étant de 1: la plupart des projets open-source ne sont pas le travail d’un groupe, et ne requièrent donc pas d’organisation. La plupart des développeurs sont Européens ou Américains (80%) et plus de la moitié sont des professionnels dans les industries de l’information et de la communication. Contrairement à une perception populaire, moins d’un tiers des développeurs sont des étudiants, et plus de la moitié sont payés pour développer leur projet ou le faire fonctionner dans leur entreprise [7]. La plupart des contributeurs à un projet open-source sont motivés au départ par le désir d’apprendre à utiliser le logiciel, et ceux qui sont payés pour le faire sont particulièrement intéressés par la possibilité de l’adapter aux besoins de leur travail. Le travail d’un développeur ou utilisateur de logiciel libre consiste à s’assurer du bon fonctionnement du logiciel, résoudre les incidents de fonctionnement, éventuellement corriger le code et proposer des changements. Les responsables d’un projet doivent en outre maintenir une distribution du logiciel, c’est à dire tenir à jour l’ensemble de fichiers nécessaires au fonctionnement et à la compréhension du programme, en assurer la diffusion et coordonner le travail des développeurs du projet. La plupart travaillent moins de 5 heures par semaine sur leur projet, mais près de 15% y passent de 15 à plus de 40 heures par semaine.

Il est difficile d’évaluer l’impact économique du logiciel libre, que cet impact se manifeste dans les emplois liés aux industries de service qui l’entoure, par son usage dans les entreprises, par les particuliers et dans l’infrastructure de l’Internet, ou par la concurrence qu’il représente pour les logiciels propriétaires. La question essentielle dans cet article n’est pas tant de déterminer l’importance des activités liées au logiciel open-source, mais de déterminer si l’existence de ce type de logiciel augmente le bien-être social, que ce soit parce que, comme le disent ses défenseurs, il est de meilleure qualité, plus innovant, plus robuste, plus flexible ou plus facile d’emploi que les logiciels propriétaires qu’il remplace, ou parce que son développement est plus rapide, mieux ciblé et mieux coordonné que le développement propriétaire (F uggetta (2003)).

La section 2.2 présente quelques exemples de logiciels open-source populaires, et définit les domaines d’application de ce type de logiciels. La section 2.3 définit plus avant les questions que ce papier se propose de traiter.

2.1 Définition de l'Open Source Initiative

Bruce Perens et Eric Raymond ont créé l'Open Source Initiative à la fin des années 90 dans le but d'encourager l'utilisation de licences libres. Ils souhaitent qu'elles se conforment aux directives énoncées dans un document écrit par Bruce Perens qui définit quels critères un logiciel doit remplir pour être inclus dans Debian, une distribution Linux. Ce document représente un consensus de beaucoup des développeurs de logiciels libres sur ce qui définit le concept de libre, et nous exposerons dans cette partie les points qui nous semblent les plus intéressants d'un point de vue économique. Ces définitions montrent bien les objectifs que beaucoup des développeurs de logiciel libre essaient d'atteindre.[3]

1. **Redistribution libre et gratuite** "La licence d'un composant d'un logiciel libre ne doit pas empêcher un contractant de vendre ou donner le logiciel sous forme de composant d'un ensemble (distribution) constitué de programmes provenant de différentes sources. La licence ne doit requérir ni redevance ni rétribution sur une telle vente."

   Cela signifie qu'une licence de logiciel libre ne peut empêcher la diffusion de ce logiciel. La licence doit permettre l'utilisation du logiciel dans d'autres applications que celles pour lequel il avait été originalement conçus. Il est important de remarquer qu'il est explicitement interdit de restreindre l'utilisation du logiciel à des applications non commerciales.

   La portée de cette clause est controversée. La General Public License ('GPL') empêche l'intégration du code libre dans un logiciel propriétaire. L'Open Source Foundation la considère quand même comme une licence libre, en s'appuyant sur le fait qu'un logiciel propriétaire pourrait faire appel à un logiciel sous licence GPL pour exécuter certaines de ses fonctions, tant que les deux logiciels restent distincts et que le code du logiciel libre ainsi exploité est mis à disposition de l'utilisateur suivant les règles de sa licence propre. Ce point est controversé car la définition de 'distinct' est soumise à interprétation, et n'a pas encore été testée devant des tribunaux.[4]

2. **Code source** "Le programme doit inclure le code source, et la diffusion sous forme de code source comme sous forme de programme compilé doit être autorisée."

3. Il est dangereux de s'aventurer dans le domaine de la définition du logiciel libre. La Free Software Foundation, par exemple, insiste sur le fait que "the Free Software movement and the Open Source movement are today separate movements with different views and goals, although we can and do work together on some practical projects." Pour les besoins de cet article, nous n'attacherons que peu d'importance à ces distinctions.

4. Cette ambiguïté se voit par exemple dans le cas de la base de données MySQL. MySQL peut être obtenue sous l'une ou l'autre de deux licences, une licence GPL et une licence commerciale. La licence commerciale, pour laquelle il faut payer, autorise son propriétaire à ne pas distribuer le code source d'applications qui s'appuieraient sur MySQL. Le site web de MySQL donne une définition très extensive des cas où une licence commerciale doit être achetée: "If you distribute a proprietary application in any way, and you are not licensing and distributing your source code under GPL, you need to purchase a commercial license of MySQL" (*http://www.mysql.com/products/licensing.html*, visité le 26 mars 2004).

C’est là l’aspect ‘ouvert’ du logiciel libre ; le programme doit être diffusé avec l’ensemble des instructions qui régissent son fonctionnement, dans un langage compréhensible par les humains. Cela constitue la principale différence avec un autre ensemble de logiciels gratuits, les ‘freewares’ qui sont distribués gratuitement mais sans le code source – Internet Explorer, le navigateur Internet de Microsoft, Eudora Light, un client de messagerie de Qualcomm ou Adobe Acrobat Reader en sont des exemples bien connus.

3. **Applications dérivées** “La licence doit permettre les modifications et les applications dérivées, et elle doit permettre à celles-ci d’être distribuées sous les mêmes termes que la licence du logiciel original.”

   Le logiciel doit donc être librement modifiable, et ses modifications doivent pouvoir être distribuées de façon libre. Il s’agit là d’une vraie limite : des fabricants de logiciels propriétaires pourraient vouloir ouvrir leur code afin que les usagers puissent signaler ses défauts, tout en évitant que des versions dérivées soient mises sur le marché.

   Cette clause incite les responsables de projets à intégrer les améliorations proposée par un utilisateur sera prise en compte, pour éviter qu’une version améliorée ne soit livrée par d’autres, plus attentifs aux suggestions des usagers. Cette clause ne requiert pourtant pas que cette nouvelle version ait la *même* licence que l’originale, quoique cette clause supplémentaire fait bien partie de la licence GPL. En particulier, la Berkeley Software Distribution (‘BSD’) permet même que la nouvelle version soit propriétaire.

4. **Intégrité du code source de l’auteur** “La licence peut défendre de distribuer le code source modifié seulement si elle autorise la distribution avec le code source de fichiers correctifs (patch files) destinés à modifier le programme au moment de la génération. La licence doit autoriser explicitement la distribution de logiciels générés à partir de code source modifié. Elle peut requérir que les applications dérivées portent un nom ou un numéro de version différents de ceux du logiciel original.”

   Cette clause donne la possibilité pour un auteur de garder un droit de regard sur ce qui adviendra de son travail : il peut demander à ce qu’une modification de son code par quelqu’un d’autre soit clairement signalée comme telle. Cela lui évite notamment de se voir attribuer des erreurs qui ne sont pas les siennes. Il protège ainsi sa réputation en tant que programmeur, d’autant plus importante que se construire une réputation est une motivation majeure pour les programmeurs libres. Surtout, cette clause clarifie les responsabilités pour le développement du logiciel – il devient aisé de savoir à qui s’adresser pour régler un problème. Elle a fait l’objet d’un débat lorsque la licence de TEX, un logiciel de mise en page, a été écrite, car cette licence requiert que la version originale du logiciel soit distribuée avec la version modifiée. Certains membres de l’équipe Debian ont considéré que cette clause

additionnelle restreignait par trop la libre modification du code, tandis que les développeurs de TEX voulaient imposer cette clause pour garantir une plus grande uniformité des versions du logiciel en circulation.

5. **Pas de discrimination à l'égard de personne ou d'entités légales**

   Une licence ne peut empêcher, par exemple, une entreprise concurrente de celle qui a écrit le logiciel d'utiliser ce logiciel. Cela conduit de nombreux utilisateurs de logiciels open source à garder leur travail sur le logiciel secret de façon à empêcher leurs concurrents d'en profiter. Mais cela garantit aussi que chacun puisse contribuer au développement du logiciel, le pool d'expertise n'étant pas limité.

   Il faut aussi remarquer que cette clause interdit des restrictions du genre "pas d'utilisation dans l'industrie de l'armement".

6. **Pas de discriminations à l'encontre de champs d'application**

   Cette clause a les mêmes origines et mêmes effets que la précédente.

7. **Licence des distributions** "Les droits attachés au programme doivent s'appliquer à tous ceux à qui il est distribué sans obligation pour aucune de ces parties de se conformer à une autre licence."

   Cette clause empêche que ceux auxquels le logiciel est re-distribué aient à faire face à des obligations supplémentaires que celles qui ont été imposées à ceux qui le redistribuent. Elle complète les clauses de non-discrimination précédentes.

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

   Cela signifie que la licence porte sur le code, non sur son expression. Il s'agit de garantir que le code puisse être 'recyclé' dans de nouveaux produits, de façon à ce qu'une innovation exprimée dans un programme puisse être exploitée dans d'autres secteurs.

9. **La licence ne doit pas contaminer d'autres logiciels**

   La licence ne peut en particulier requérir que le logiciel soit distribué uniquement avec d'autres logiciels libres.

10. **La licence doit être neutre d'un point de vue technologique**

    Il n'est ainsi pas possible d'imposer l'utilisation d'une certaine interface, d'un langage de programmation ou d'un logiciel donné, pour exploiter ou développer le logiciel. Cela limite les tentations de promouvoir ces derniers à l'aide d'une offre logicielle libre. Cela n'empêche pourtant pas de nombreuses entreprises logicielles de promouvoir leur offre à l'aide

du logiciel libre. Sun a ainsi permis que sa plate forme de programmation Java fasse partie de distributions libres afin d'encourager l'utilisation de ses produits.[5] De même, Adobe a rendu son langage de programmation de document, PostScript, libre afin d'encourager la diffusion de ses standards pour le formatage de documents, et l'achat de ses logiciels propriétaires qui offrent des fonctionnalités supplémentaires.

2.2 Les domaines d'application des programmes open-source

Le tableau 1 p.8 présente quelques uns des programmes open source connus.

Les logiciels libres se concentrent dans certains domaines d'application, que ce soit parce que ce sont ceux qui intéressent la communauté open source ou parce que ce type de développement y est particulièrement approprié :

- Les applications 'techniques' ou d'infrastructure Internet ; c'est en effet là que le besoin de faire appel à des contributions diverses dans un environnement logiciel toujours changeant rencontre une audience qualifiée (les gestionnaires du réseau).

- Peu pour l'instant dans les ordinateurs de bureau, quoique Linux soit maintenant distribué par des entreprises commerciales telles que Red Hat, dans des versions dont certains pensent qu'elles peuvent rivaliser avec Windows en terme de facilité d'utilisation.[6]

- Les applications embarquées (embedded applications), c'est à dire celles qui régissent le fonctionnement des appareils électroniques. Ainsi, Sony et Matshushita se sont associés pour développer des versions de Linux pour les appareils électroménagers, télévisions, DVD, PDA, etc.

Les développeurs de logiciels étant leurs premiers utilisateurs, il est compréhensible que les logiciels open-source soient particulièrement présents dans des domaines requérant une expertise informatique – langages de programmation, de gestion de base de donnée, serveurs. Il y a peu de logiciels open-source 'grand public', non pas qu'il n'existe pas de traitement de textes, jeux, ou systèmes d'exploitation open-source, ou que ceux-ci soient de moins bonne qualité que les programmes propriétaires équivalents, mais parce que ceux-ci sont difficiles à utiliser pour des novices. La documentation des logiciel libre n'est en général compréhensible que pour des publics déjà averti, et leur installation nécessite une bonne connaissance du système d'exploitation

5. Java est fourni gratuitement aux développeurs, mais n'est pas open source. Les versions libres dérivées doivent adopter d'autre noms et ne sont pas certifiées par Sun. Sun encourage l'usage de Java afin d'accroître la valeur de ses produits sur lesquels tournent les logiciels développés avec ce langage de programmation.

6. Un état des lieux en ce qui concerne l'utilisation de Linux sur les ordinateurs de bureau, écrit par des membres de la communauté open source, peut se trouver à `http://www.osafoundation.org/desktop-linux-overview.pdf`.

Projet	Description	Indicateurs	License et organisation
Linux	Système d'exploitation, équivalent de Windows ou MacOS.	Utilisé par près de 30 millions de personnes. 25% du marché des systèmes d'exploitation pour serveurs web.	Distribué sous licence GPL. Développé sous le leadership de Linus Torvalds par plus de 1,000 développeurs.
Apache	Serveur web.	Apache a une part importante du marché des serveurs web, essentiels à la gestion de l'infrastructure Internet des entreprises,	Licence de type BSD. Développement contrôlé par un groupe de développeurs regroupés dans une association, l'Apache Foundation.
SendMail	Logiciel de gestion du courrier électronique	Un des premiers programmes pour le routage du courrier électronique, il est toujours le plus utilisé dans ce domaine.	Sous licence BSD, développé par Eric Allman. Eric Allman a crée sa propre compagnie pour vendre une version propriétaire proposant plus de fonctionnalités.
FreeBSD, OpenBSD, NetBSD	Systèmes d'exploitation, versions d'Unix	Unix a été développé par AT&T avec l'aide de développeurs libres, jusqu'à ce que AT&T décide de l'exploiter de façon propriétaire.	Sous licence BSD. Les différents groupes de développeurs poursuivents des objectifs différents mais le développement reste synchronisé.
Perl, Ruby	Langages scripts de programmation	Essentiels pour la présentation de données dynamiques sur Internet	Développement dirigé respectivement par Larry Wall et Yukihiro Matsumoto, sous license Artistique ou GPL
TEX	Logiciel de composition	Couramment utilisé dans la communauté académique, particulièrement adapté aux mathématiques	License de type BSD, développement contrôlé par le créateur, Knuth, puis par le groupe des usagers de TEX

TAB. 1: *Programmes open source connus*

utilisé (Nichols-Twidale(2003)). Il n'y a que peu de raison pour des développeurs de logiciels libres d'automatiser le processus d'installation ou d'expliquer le fonctionnement du logiciel à des personnes qui ne peuvent contribuer à son développement.

Pour illustrer cet article, on s'attachera plus particulièrement au cas de TEX. TEX est un logiciel de mise en page particulièrement adapté aux mathématiques et couramment utilisé dans la communauté académique. C'est un cas intéressant car c'est un logiciel sous licence BSD dont le développement a commencé à la fin des années 70 et dont la communauté de développeurs était tournée dès le départ vers l'usager final sans expertise informatique. Il s'agit donc d'un cas atypique par rapport aux études de cas faites jusqu'ici sur Linux ou Apache (voir Gaudeul (2003))

TEX fournit un exemple intéressant des problèmes d'interface auxquels font face les logiciels libres. L'apparition du logiciel Word pour Windows de Microsoft en 1989 a détourné une partie des utilisateurs de TEX, fondant son succès sur l'élaboration d'une interface graphique d'usage agréable. Le développement d'une telle interface demande un long travail de conception et des ajustements permanents que les projets open source ne sont généralement pas prêts à prendre en charge, ces détails accroissant peu la valeur du logiciel pour des personnes qui l'utilisent déjà en expert. L'absence de support centralisé pour les usagers et de moyens publicitaires est une raison supplémentaire pour les difficultés des programmes libres sur le marché grand public. La diffusion de TEX dans les universités s'est faite par le bouche à oreille et uniquement grâce au travail de volontaires pour adapter le logiciel à leur système et former les usagers.

D'autres raison semblent expliquer le succès de l'open-source en ce qui concerne les applications embarquées. Choisir un logiciel open-source favorise la collaboration entre les entreprises, qui ne risquent plus de voir leurs contributions appropriées par d'autres. Cela permet aussi d'économiser sur les coûts de développement, les entreprises n'ayant plus à poursuivre leurs développements en parallèle. La section 3 explique plus avant comment les motivations des usagers, sponsors et développeurs de logiciels open-source se traduisent en un ensemble de caractéristiques communes pour les logiciels libres.

2.3 Problèmes à résoudre

Les questions que l'on cherchera à résoudre se divisent en deux parties : Premièrement, comment fonctionnent les projets open source : pourquoi attirent-ils des contributions ? Pourquoi ne semblent-ils n'avoir de succès que dans certains domaines ? Il s'agit là de faire une analyse des incitations économiques des développeurs, usagers et sponsors de l'open source, car celles ci déterminent quels logiciels seront développés en open source. En second lieu, quel est l'impact des logiciels open source sur le bien-être global ? Il s'agit notamment de déterminer si ce type de licence encourage le développement de logiciels plus sûrs, plus fiables, de meilleure qualité et plus innovants que les logiciels développés de façon classique sous licence propriétaire. Cela

permettra d'analyser quelle devrait être la politique des pouvoirs publics face à ce phénomène qui opère en grande partie en dehors de toute régulation ou intervention gouvernementale.

3 Logiciel libre et incitations

Cette partie examine les incitations des programmeurs, utilisateurs et "sponsors" de projets open-source. Cela nous permettra d'éclairer les déterminants du choix de licence pour le développement d'un logiciel.

3.1 Incitations des programmeurs

Les licences open source obligent les programmeurs à renoncer à presque toute la protection sur le travail intellectuel exprimé au travers de l'écriture d'un programme. Dans ce cas, si l'on en croit la logique qui amène les gouvernements à encourager l'innovation en conférant des brevets et licences aux inventeurs, il n'y aurait aucune motivation à développer du logiciel libre. Et pourtant, les logiciels libres existent…

Différents auteurs ont proposé différentes explications à cette énigme. Pour Benkler (2001), les programmeurs tirent des bénéfices non monétaires de leur travail. Il y a le plaisir de la programmation, le désir de se rendre utile, et aussi le désir de se faire accepter dans une communauté de développeurs considérée comme prestigieuse. Cette explication n'est pas très satisfaisante: après tout, ces satisfactions peuvent être obtenus aussi en travaillant dans une entreprise propriétaire, où en plus le programmeur serait rémunéré. Après tout, la plupart des chercheurs sont rémunérés pour leur travail et y trouvent du plaisir, se sentent utiles en le faisant, et sont heureux de faire partie et d'être reconnus par leur communauté scientifique.

Justin Johnson (2003) reprend l'argument de Benkler, de façon plus subtile. Il considère un modèle dans lequel les développeurs répondent à la fois aux incitations monétaires et aux incitations non monétaires. Il montre que dans ce cas-là les incitations monétaires peuvent en quelque sorte "polluer" les incitations non monétaires. Une entreprise commerciale garde un droit de propriété sur le travail de ses développeurs, et s'en servira, au moins dans certains cas, d'une façon différente de celle qu'ils auraient eux-même choisie. Cela peut entraîner une diminution des incitations. Vu ainsi, le logiciel libre serait une façon plus efficace de tirer profit des incitations non monétaires des programmeurs.

Certains autres auteurs argumenter qu'un travailleur bénévole, dont l'objectif est de contribuer à une augmentation du bien-être social, préfèrera travailler sur un logiciel non commercial. En effet, contribuer volontairement et gratuitement à un logiciel commercial ne sera pas efficace si son but est de contribuer au bien-être social : tout travail ainsi effectué ne servira qu'à réduire les besoins de travail rémunéré de l'entreprise, sans que pour autant le total du travail effectué

augmente. La logique de l'entreprise est de maximiser ses profits, et sa contribution au bien-être social sera la même qu'elle ait reçu des contributions gratuites ou non. Ainsi, un développeur qui veut contribuer au bien-être social ne pourrait le faire qu'en open source. Il faut toutefois remarquer que le même raisonnement devrait tenir pour d'autres secteurs d'activité. Un ingénieur qui voudrait contribuer au bien-être social en travaillant bénévolement à l'amélioration des voitures voudrait participer à des projets de voitures libres. Comme on ne voit pas de projets libres dans d'autres secteurs, cette explication du développement des logiciels libres ne peut donc s'appliquer que si l'on pense que les développeurs et programmeurs sont plus altruistes que d'autres ou que l'altruisme trouve dans le développement de logiciel une expression particulièrement aisée.[7] Ce contre argument est validé par le fait que les études empiriques montrent que le bénévolat n'est pas la principale motivation pour la plupart des développeurs.

Une explication populaire est celle de 'l'homme (ou la femme) providentiel(le)'. À partir du moment où une personne décide de mettre son travail à disposition des autres, mais qu'il pose comme condition que ces autres fassent de même, et que ce premier apport est nécessaire à tous, alors il y aura un effet boule de neige. Les autres développeurs ne travailleraient en libre que parce que c'est là la règle établie au départ, et qu'il n'y a pas de possibilité de s'en libérer. À l'origine du logiciel libre, se trouverait donc un accident.

D'autres explications font appel à des motivations de nature économique. Ainsi, Lerner et Tirole (2002) soulignent le rôle de signal sur le marché du travail que joue la participation à un projet open source. D'une part, elle prouve la capacité à comprendre et à contribuer à un projet logiciel, mais aussi, du fait d'un système d'hiérarchie des contributions, elle permet d'assigner un 'rang' à un développeur. Ce rang et le prestige du projet permettent de différencier les développeurs d'une façon que le travail de programmation en entreprise ou leur formation en école, ne permet pas. On rencontre ainsi des programmeurs qui ont été embauchés par des entreprises produisant des logiciels propriétaires ou qui ont reçu un financement de capital risque grâce à leurs contributions au logiciel libre ! De plus, même en l'absence de monétarisation différée des contributions, le prestige (qui est conditionné à une visibilité claire des contributions individuelles) est un incitant puissant. Peyrache (2002) propose un modèle pour analyser le choix des projets sur lesquels des agents motivés par leurs perspectives de carrière vont travailler. Suivant leur qualité, ils choisiront des projets sur lesquels il y a plus ou moins de compétition.

Enfin, von Hippel (2002) explique que les communautés open source forment des réseaux d'usagers qui peuvent fonctionner de façon totalement indépendante de tout motivation monétaire ou bénévole ; les développeurs libres utilisent le logiciel à des fins personnelles, et n'ont pas envie d'investir le temps et l'argent nécessaire pour faire une exploitation commerciale de leurs contributions. Il leur est par contre facile de fournir leur code open-source, sans qu'il y ait besoin

7. On peut aussi se demander si un individu parfaitement altruiste choisirait l'écriture sans rémunération de logiciels plutôt que le volontariat aux "Restos du cœur !

d'introduire d'incitations particulières à le faire au-delà de l'esprit d'entraide entre usagers. Ce type d'explication est valide pour un grand nombre de programmes Internet ; dans le cas du serveur Web Apache, des administrateurs réseaux qui ne font pas partie d'entreprises logicielles forment des réseaux spontanés d'entraide et de support. De même, le logiciel de composition TEX était à l'origine destiné à satisfaire uniquement les besoins d'un développeur, Donald E. Knuth, qui s'intéressait également à la typographie, mais pour qui ce logiciel n'a jamais constitué un objectif de carrière prioritaire. Ceci dit, la plupart des développeurs de logiciels libres travaillent pour des entreprises qui cherchent à obtenir des profits à partir de leur travail. Il serait intéressant d'adapter le travail de von Hippel à cet environnement.

Gaudeul (2003) revient sur les raisons pour lesquelles un développeur peut choisir de rendre son travail public et recourir au travail des développeurs libres au lieu d'essayer de le mettre sur le marché : choisir une licence libre permet de réduire les coûts de développement et de maintenance du logiciel. Les licences libres seraient ainsi une réponse au manque de développeurs sur le marché du travail ; les employer à rendre un logiciel commercialisable n'est pas rentable et il est ainsi préférable d'en délivrer une version inachevée que la communauté libre se chargera de rendre exploitable.

Tout cet ensemble de motivations fait que les logiciels open source partagent des caractéristiques et limites communes : ce sont des logiciels faits pour des experts, qui ont un langage commun hautement spécialisé et ne se préoccupent pas (ou peu) de faciliter l'usage de leur logiciel par les non-initiés. Pour des projets relativement simples, ces experts ne sont également que peu contraints par des disciplines de groupe, car ils sont capables d'utiliser et d'adapter le logiciel sans l'aide de la communauté. Cela explique le peu de discipline dans certains projets, le phénomène du 'forking' (développement séparé de la branche principale) étant relativement fréquent.

Néanmoins, pour des projets complexes tels que Linux, il est de l'intérêt de chacun de se conformer à la logique du groupe car personne ne peut à lui seul influer sur son développement global. Un système de sanctions implicites décourage ceux qui ne tiennent pas compte du consensus des développeurs sur la direction du projet ; faire bande à part prive un développeur de support. Une grande part du travail d'un développeur consiste donc à convaincre le groupe du bien fondé de ses propositions. Il existe aussi une motivation à propager le logiciel si celui-ci bénéficie d'effets de réseau, et cela explique que de nombreux projets libres se préoccupent de rendre leur logiciel plus facilement accessible aux usagers et de faciliter les contributions à son développement. Il leur est difficile néanmoins de s'accorder sur des objectifs communs, surtout lorsqu'il s'agit de rendre le logiciel populaire. Il manque aux projets libres les signaux du marché qui indiquent aux projets commerciaux s'ils vont dans le bon sens. Il n'y a pas de prix, donc pas d'évaluation de la valeur du logiciel, et l'importance du marché pour le logiciel ne peut que

difficilement être évaluée dans la mesure où l'acquisition et l'utilisation du logiciel ne fait pas l'objet de transactions explicites.[8]

3.2 Incitations des entreprises

Une part importante des contributions à des projets open source est effectuée par des entreprises commerciales, qui payent des programmeurs pour contribuer au développement de ces programmes. IBM est un exemple, mais également Hewlett-Packard ou Sun et son langage de programmation Java[9]. Ces entreprises vendent en effet des produits complémentaires aux logiciels open source : matériel tels qu'ordinateurs et serveurs qui pourront être vendus plus chers si les logiciels livrés avec sont gratuits, services tels qu'expertise en programmation ou en utilisation des logiciels. Elles utilisent également les logiciels open source dans leurs stratégies commerciales : Oracle promeut ses logiciels de gestion de bases de données en montrant qu'ils fonctionnent mieux avec Linux que ceux d'IBM.

Livrer le code à la communauté open source peut également être une manière de profiter de ses ressources en temps et expertise. Netscape a ainsi libéré le code de son navigateur Internet en 1998 de façon à encourager son développement, ses incitations à le faire étant renforcées par sa perte de part de marché face à Internet Explorer de Microsoft. D'autres entreprises espèrent rendre leur produit plus sûr et de meilleure qualité à partir du principe que plus il y aura de personnes qui pourront vérifier le code, meilleur il sera. Elles gardent la possibilité d'exploiter leur code de manière commerciale en faisant valoir leur expertise unique sur leur propre produit.

Pour des projets importants d'un point de vue stratégique, comme le sont par exemple Linux ou Apache les entreprises veulent influencer le développement du projet en plaçant certains de leurs employés au sommet de leur hiérarchie. C'est ainsi que de nombreux leaders de projets open source importants travaillent pour de grandes sociétés d'informatique.

Dans la plupart des cas, donc, une stratégie 'open source' joue en annexe d'une stratégie de développement plus globale. Il y a pourtant des entreprises logicielles qui mettent le logiciel libre au cœur de leur stratégie. Elles y trouvent un avantage en terme de vitesse de développement, profitant de la contribution de personnes qu'elles n'auraient jamais les moyens de payer. Cet avantage compense la perte d'un avantage compétitif, celui du contrôle de leur moyen de production, et l'entrée facilitée de concurrents (voir Baake & Wichman (2003)).

D'autres entreprises qui se chargent de rendre le logiciel libre accessible aux usagers finaux, telles que Red Hat pour Linux, ont intérêt au succès du logiciel et favorisent donc son développement et sa diffusion. Pour les logiciels sous licence BSD, ces entreprises font des interfaces en

8. Les distributions commerciales de Linux, telles que Red Hat ou Mandrake, sont en compétition pour répondre aux besoins des usagers, mais toute innovation de l'une peut être librement appropriée par l'autre, ce qui réduit les motivations à innover.

9. Java n'est que partiellement un logiciel libre, mais nous négligerons ces différences mineures pour cet exposé.

ajoutant des fonctionnalités qui ne sont pas reversées au fond commun. Elles contribuent à la diffusion du logiciel, mais participent peu au développement. Pour les logiciels sous licence GPL, cette stratégie est vouée à l'échec, tout développement devenant propriété commune à partir du moment où il est diffusé. Le modèle d'affaire réside alors dans le support fourni aux acheteurs de la distribution commerciale.

3.3 Incitations des usagers

Pourquoi 'achète-t-on' du logiciel libre ? La première raison est qu'on peut souvent l'acquérir sans payer. La comparaison prix-bénéfice n'est pourtant pas si simple, car sinon, il n'y aurait plus de logiciel propriétaire. Les logiciels libres sont en général plus difficile à installer que les logiciels propriétaires, moins facile à apprendre à utiliser et n'offrent pas d'interface commode avec l'usager.[10]. Ces considérations sont peu importantes pour l'usager typique d'un logiciel open source qui au contraire peut se sentir limité par l'impossibilité de faire des changements dans le logiciel ou l'adapter à son usage.

D'autre part, la qualité d'un logiciel open source peut-être supérieure à celle d'un logiciel propriétaire si cette qualité est évaluée suivant les critères de développeurs : nombre d'erreurs dans le code et vitesse de résolution des problèmes (voir Kuan (2002)). Certains prétendent qu'il est moins sujet au piratage, le code ouvert permettant à plus de personnes d'identifier les défauts. Bien sûr, ceci aide les attaquants autant que les défenseurs. Anderson (2002), en s'appuyant sur les théories de l'évolution de la qualité des logiciels, montre que dans l'ensemble, un système fermé pourra arriver à être aussi sûr qu'un système ouvert.

D'autres raisons à long terme entrent en ligne de compte : choisir un système logiciel est un investissement important en terme d'apprentissage et de formation, et un usager qui choisit un logiciel manifeste ainsi sa confiance en la communauté ou l'entreprise qui le soutient. Si le logiciel libre est plus innovateur que le logiciel propriétaire, s'il est plus stable ou plus pérenne, alors la balance penche de son côté même si à court terme son adoption requiert des adaptations coûteuses.

En tout état de cause, il semble que l'aspect libre du logiciel importe moins que pour la plupart des utilisateurs que son faible coût et ses caractéristiques propres.

3.4 Choix de licence

Le choix de licence (propriétaire ou libre, et si libre, GPL ou BSD) tient compte des motivations des usagers, développeurs et sponsors, et restreint leur comportement en conséquence. Les

10. Selon certains experts, les logiciels serveurs de Microsoft pour PME peuvent être installés trois fois plus vite que les logiciels Linux correspondant (`http://news.com.com/2100-1012-5068922.html`, visité le 9 septembre 2003).

licences libres ne sont pas des licences permissives, elles imposent un certain nombre d'obligations aux développeurs et visent à garantir la libre circulation et la libre modification du code. Mais elles permettent des variations. En pratique, les licences restreignent l'utilisation du logiciel, à des degrés divers : la plus restrictive est la licence GPL qui ne permet pas l'utilisation du code dans un logiciel qui n'est pas lui-même GPL. La licence BSD ne met pas de restrictions sur l'utilisation du code, mais ne permet pas que la licence du logiciel original soit modifiée (voir points 1 et 3 de la section 2.1). Finalement, mettre son logiciel dans le domaine public signifie que n'importe qui peut se l'approprier et y appliquer sa propre licence.

Le choix de licences plus ou moins restrictives dépend des objectifs stratégiques des 'entrepreneurs'. Lerner et Tirole (2002) montre l'influence des caractéristiques du profil des utilisateurs du logiciel.Lorsque un programme mais que celui-ci a un grand intérêt pratique pour des usagers autres que les développeurs, les licences restrictives de type GPL seront utilisées. Ceci évite l'expropriation du programme par un logiciel propriétaire qui n'en différerait que par une interface plus agréable. Au contraire, les programmes intéressants pour ceux qui les construisent ont des licences plus permissives. Les programmes sont utiles aux développeurs n'ont pas besoin de licences restrictives, car toute tentative de mise sur le marché serait vouée à l'échec, une version libre équivalente étant déjà, et effectivement, disponible. Une licence restrictive de type GPL réduirait trop la latitude des développeurs car ceux-ci veulent pouvoir utiliser côte à côte du code open source et des logiciels propriétaires. Les licences de type BSD sont ainsi destinées aux logiciels 'professionnels' tandis que les licences GPL sont destinées aux logiciels d'usage moins spécialisé.

Une autre perspective sur le choix de licence est offerte par le cas de TEX. Du fait de sa licence BSD, TEX a dès le départ fait l'objet d'applications propriétaires qui n'ont pas directement profité au développement du logiciel. La communauté libre ne s'est organisée que tardivement pour offrir une distribution accessible au plus grand nombre, mais lorsqu'elle l'a fait, cela a rendu la plupart des applications propriétaires obsolètes. L'exploitation commerciale ne semble pas avoir découragé le développement libre, car les publics utilisant la version libre (éditeurs, compositeurs et autres professionnels) étaient bien distincts, et avaient d'autres objectifs, que les publics utilisant les versions propriétaires (particuliers). La compétition avec des logiciels propriétaires dans le même domaine est intéressante à observer, le logiciel ayant emprunté nombre de ses fonctionnalités à des logiciels propriétaires existant à l'époque où il a été développé, mais ayant également été imité et ayant eu son code emprunté par les logiciels propriétaires qui ont suivi. Ce cas souligne que bien souvent, les logiques du développement propriétaire et libre ne vont pas l'une contre l'autre; choisir une licence BSD peut contribuer à rendre le logiciel plus populaire, et attirer l'attention d'entreprises qui seront prête à sponsoriser son développement.

Pour conclure, certains éditeurs donnent à l'utilisateur le choix entre plusieurs licences. Par

exemple, Red Hat, un distributeur Linux, propose le logiciel Cygwin qui permet de faire fonctionner des programmes écrits pour Linux sur des PC “Windows”, à la fois sous une licence GPL et sous une licence privée, qui permet de l’intégrer dans une application propriétaire. La base de données MySQL est aussi disponible dans une version propriétaire et une version GPL. Nous ne connaissons pas d’analyse économique de ces licences duales, qui posent des problèmes intéressants.

4 Logiciel libre et politique publique

Les gouvernements doivent-ils encourager le développement et l’adoption de logiciels open-source ? Quatre types d’action en faveur du logiciel libre peuvent être envisagées par la puissance publique : encourager l’utilisation de logiciels libres dans les administrations publiques, subventionner le développement des logiciels libres, demander à ce que les résultats de la recherche publique soient mis sous licence libre, et finalement clarifier le cadre juridique notamment autour de la licence GPL.

Pour évaluer le bien fondé de ces actions, il est nécessaire de se placer tour à tour du point de vue du producteur et du consommateur. Le gouvernement peut-il influencer les choix du consommateur de façon à accroître leur bien-être ? D’autre part, en amont, favoriser le développement de logiciels open-source risque-t-il de décourager l’innovation en ce domaine comme le proclament des associations de développeurs de logiciels propriétaires ? La section 4.1 examine les options de politique publique et les conditions sur le marché du logiciel pour répondre à la première question. La section 4.2 répond à la seconde question en recadrant le débat dans le contexte de l’économie de la propriété intellectuelle et de l’innovation. La section 4.3 conclut sur les problèmes juridiques liés aux logiciels libres, qui ne bénéficient pas encore d’un cadre légal reconnu.

4.1 Politique publique et bien-être du consommateur

Deux types de raisonnement peuvent être employés pour justifier l’achat de logiciel libre par les administrations. Tout d’abord, les responsables d’achat peuvent être convaincus que le logiciel libre représente un meilleur produit, une fois considéré tous ses avantages et ses coûts. Dans ce cas, il n’est pas contestable qu’ils doivent choisir le logiciel libre, mais cela ne nécessite pas une politique publique : n’importe quel consommateur agissant dans son propre intérêt choisirait dans ce cas le logiciel open-source.

Il faut donc pour justifier un encouragement au logiciel libre penser que les acteurs économiques ne sont pas capables de se coordonner sur le type de logiciel qui est préférable pour l’ensemble : il y aurait un effet d’inertie sur le marché qui empêcherait, au cas où le logiciel libre

est réellement supérieur, de l'adopter. Cette analyse passe par la prise en compte d'effets de réseau, un logiciel ayant d'autant plus de valeurs que d'autres l'utilisent, ou n'étant développé que s'il est utilisé par beaucoup. Un logiciel propriétaire établi ne pourrait être délogé, même s'il est inférieur à son alternative libre, et il y faudrait donc un encouragement de l'Etat. Les effets de réseau existent également pour les logiciels libres, un logiciel libre dominant pourrait donc lui aussi être inférieur à son alternative propriétaire. Favoriser systématiquement les logiciels libres ne peut pas être justifié sur cette base.

Si le bien-être immédiat du consommateur ne justifie pas l'action publique, il se pose par contre un problème de politique économique si des administrations achètent des logiciels libres parce qu'elles espèrent favoriser la création d'une industrie logicielle nationale ou bien créer une nouvelle dynamique de l'innovation dans ce domaine. L'expérience de la politique économique dans le domaine des logiciels montre bien toute la difficulté de réaliser ce type d'objectif.

Une erreur souvent faite, liant effets de réseau et logiciel libre, est de penser que le code ouvert des logiciels libres amène naturellement à ce que les standards sur lesquels ils sont fondés soient également ouverts. Cela amènerait à justifier l'encouragement au logiciel libre afin de favoriser l'adoption de standards. En fait, le cas de TEX notamment montre qu'il peut exister une tension entre le caractère libre du logiciel, et la nécessité pour ce format d'échange de document d'adhérer à un standard. C'est pour cela qu'une licence assez restrictive et un système de certification des distributions ont été mis en place pour garantir que tous les systèmes TEX soient compatibles entre eux. Cela n'a pas empêché des systèmes inspirés de TEX, mais n'adhérant pas à ses standards, d'émerger. De fait, loin de respecter un standard, les développeurs de TEX consacrent beaucoup d'efforts à mettre le système TEX aux normes édictées par Adobe, qui fait des logiciels propriétaires, pour produire des documents au format PDF, et aux normes édictées par le World Wide Web Consortium ('W3C') qui contrôle entre autre les spécifications du standard XML ('Extensible Markup Language') pour la description de documents. D'autre part, un standard 'ouvert', au sens que tout le monde puisse le changer, n'est pas désirable d'un point de vue de politique publique, car il n'y aurait aucun moyen d'imposer un accord sur ses spécifications. Un standard ouvert désirable se limite donc à être un standard dont les spécifications sont accessibles à tous.

4.2 Politique publique et incitation à l'innovation

Le gouvernement définit le cadre juridique dans lequel se fait la production de logiciel. Dans le cas des biens intellectuels, comme les logiciels, des droits de propriété sont aussi créés par la puissance publique. L'objectif est de créer des incitations à innover en accordant des droits sur l'utilisation d'une invention à celui qui en est à l'origine, de façon qu'il puisse réclamer paiement à ceux qui utilisent son innovation.

Définir des droits de propriété n'est pourtant pas suffisant pour l'efficacité. Il faudrait aussi s'assurer que les marchés où ces droits sont échangés soient compétitifs. Or on ne peut accorder des droits de propriété intellectuelle et avoir des marchés efficaces. En effet, un marché efficient se caractérise par une égalité entre coût marginal, bénéfice marginal et prix. Or le coût marginal d'une idée est nul : une fois qu'elle a été générée, il ne coûte rien de la réutiliser. Ainsi, un marché de l'innovation sera nécessairement inefficace : de façon à encourager l'innovation, il faut garantir à l'innovateur la possibilité de restreindre l'usage de son innovation à ceux qui peuvent le payer, quel que soit le prix qu'il choisisse.

De façon plus technique, il semble donc une contradiction fondamentale entre l'efficacité dynamique qui requiert que l'on octroie des droits de monopole pour favoriser l'innovation et l'efficacité statique qui demanderait des marchés compétitifs. En fait, la situation est plus compliquée : même l'efficacité dynamique peut souffrir de trop de protection qui limite la capacité de nouveaux inventeurs d'améliorer sur une première innovation.

Pour balancer toutes ses considérations, les gouvernements disposent de plusieurs instruments. Les brevets protègent les innovations utiles et ayant requis un véritable travail de recherche, tandis que le copyright donne une protection plus faible. L'expression d'une idée sera protégée par le copyright, mais pas son application par d'autres. Enfin, les secrets de fabrication ne sont protégés que dans la limite où ils ne sont pas percés. Ayant déterminé l'étendue de la protection, il faut aussi déterminer sa durée. Une durée trop réduite réduit les incitations à innover, une durée trop longue accroît le coût total de l'inefficacité statique et peut pénaliser les inventions postérieures. Cela est particulièrement important dans le cas où une innovation peut servir de base aux innovations futures dans un domaine, auquel cas il peut être nécessaire de limiter la durée de la protection.

Les logiciels entrent dans le domaine du copyright ; celui-ci empêcherait un concurrent de copier le logiciel, mais lui permettrait de reimplémenter les idées en reécrivant le code. En conséquence, les entreprises de logiciels propriétaires protégent leurs innovations en ayant recours au secret de fabrication ; c'est pourquoi, elles ne distribuent leur programmes que sous forme compilée. À l'opposé, les développeurs de logiciels open-source sous license GPL détournent la logique du copyright pour empêcher une exploitation commerciale.

Le cas de TEX permet de bien comprendre la complexité de la conséquence du choix du mode de licence. Il a été développé avec des fonds publics (National Science Foundation aux Etats-Unis). Sa licence, ou plutôt ses licences, assignent des droits différents aux usagers-développeurs suivant que le code fait partie du noyau du programme (sous copyright de Donald E. Knuth), est un élément des applications qui exploitent ce noyau (sous licence BSD) ou fait partie des infrastructures qui lient chacun des programmes élément de ce système logiciel (les distributions sont sous licence GPL). Il n'y a donc pas de licence 'passe-partout' qui s'applique à l'ensemble

des logiciels composant la distribution de ce système open source.

Ainsi donc, l'organisation open source est un système complexe de droits et de privilèges qui ne se laissent pas traduire aisément en un système juridique rigide. Ceux qui assurent la gestion du projet, qu'ils en soient à l'origine ou qu'ils gèrent la distribution du logiciel, choisissent dans quel cadre juridique des contributions seront acceptées. Celui qui est à l'origine du projet détermine les limites des droits de ceux qui développent le logiciel, qui n'ont plus que le choix de participer ou non. Celui qui rassemble les contributions dans un ensemble cohérent qui sera distribué au public peut choisir d'en exclure certaines suivant leur licence. Un développeur qui veut voir son logiciel utilisé devra donc se conformer aux 'guidelines' de cette distribution.[11] Il n'est ainsi difficile de déterminer si les licences ainsi choisies lors du développement de TEX étaient optimales d'un point de vue du bien-être social, mais la plupart des développeurs semblent bien informés des conséquences de leur choix, et la logique qu'ils suivent semble raisonnée.

TEX pour l'essentiel est sous licence BSD, et cette licence ne pose généralement pas de problème aux entreprises de logiciels propriétaires. Leurs principales objections sont à l'encontre de la licence GPL. Beaucoup de chercheurs en informatique, dont la recherche est financée sur fonds publics, publient des programmes sous licence GPL. Ces entreprises se plaignent qu'elles ne peuvent pas se servir de cette recherche, et intégrer le code dans leurs programmes. Schmidt et Schnitzer (2002) font une analyse agrémentée d'exemples de ce problème. Ils concluent que bien que le prix nul du logiciel libre soit efficient ex-post, cela décourage les innovations 'majeures' – celles qui requièrent un véritable travail de recherche et développement. S'il fallait choisir un type de licence pour la recherche publique, la licence BSD devrait être préférée pour permettre les applications propriétaires. Dans l'ensemble, il n'y aurait pas de raisons de subventionner le développement libre, celui ci se produisant de façon spontanée dans les domaines où il est adapté, et n'étant pas adapté aux domaines où il ne se produit pas naturellement. Le rôle de l'Etat devrait se limiter à encourager l'adoption de standards ouverts pour décourager la formation de monopoles, notamment lorsqu'il s'agit de logiciels de gestion des infrastructures publiques essentielles, dont Internet fait partie.

4.3 Un cadre juridique pour le logiciel libre

Il manque de références pour s'assurer de la viabilité juridique de la licence GPL. Celle-ci a été contestée aux Etats-Unis parce que bien qu'invoquant la protection du copyright, elle permet à chacun de copier et distribuer sans limite un logiciel sans avoir à demander l'autorisation de l'auteur, alors que la loi américaine limite la copie à un exemplaire. D'autre part, la plupart des

11. La plupart des développeurs veulent voir leur logiciel utilisé, non pas seulement parce que cela prouve la qualité de leur travail, mais aussi parce qu'ils ne recevront d'aide dans le développement du logiciel que s'il est utilisé par beaucoup, les utilisateurs devenant naturellement des développeurs.

pays européens ne permettent pas à l'auteur d'un logiciel de se désister de toute responsabilité pour les conséquences de l'utilisation du code qu'ils ont écrit, ce qui pourrait décourager plus d'un de contribuer à des projets libres. Il y a donc un grand besoin de clarification du cadre légal dans lequel se place la production open-source, les incertitudes à ce propos freinant le travail des développeurs. Ceci dit, il n'est pas clair dans quelle mesure cette clarification doit être de la responsabilité de la communauté des développeurs et de la responsabilité des pouvoirs publics.

Il y a d'autre part un débat en cours à propos de la brevetabilité du logiciel, et la résolution de ce débat aura des conséquences importantes pour le logiciel libre. Il y a, nous pensons, un accord quasi-général parmi les économistes sur la théorie de la protection par les brevets. En revanche, les ordres de grandeur des effets positifs et négatifs d'une protection accrue sont très contreversés. En conséquence, le débat sur les bénéfices et les coûts de la brevetabilité sont loin d'être résolus, d'autant plus que leur résolution dépend aussi de l'estimation que l'on a de la capacité des agences de brevets à se réformer. [12]

Il est clair que la présence de brevets gênent le développement de logiciels libres, dans la mesure où les projets Open Source n'ont souvent pas d'existence légale et souvent pas de fonds. D'autre part, si l'on estime que les brevets, sous une forme ou une autre, ont un rôle à jouer dans la régulation de l'industrie du logiciel, il n'est pas clair dans quelle mesure celle-ci doit être adaptée pour tenir compte de ces contraintes - d'autant plus que certaines companies commerciales tirent leur revenu du logiciel libre.

5 Conclusion

Le logiciel libre est un phénomène économique très intéressant, qui semble pouvoir être bien pris en compte par la théorie économique des incitations, mais bien du travail d'analyse reste à faire. Les économistes ont beaucoup de contributions à apporter à l'analyse de ces phénomènes que certains pensent à priori comme non économiques. On remarque d'ailleurs qu'une fois ces analyses effectuées, elles sont bien acceptées par la communauté open source.

Ce papier a analysé les motivations des acteurs dans une organisation libre, et a appliqué cette analyse pour étudier les options de politique publique face au phénomène open-source. Nous sommes donc passé directement des motivations des acteurs à des recommandations pour la structuration légale et la gestion publique du système sans entrer dans la complexité des relations entre acteurs dans le système. Il manque donc à notre analyse une étude de la façon dont ce système complexe d'agents autogérés se régule de lui-même. Il s'agit alors d'entrer dans le

12. Tirole *et al.* (2003) propose une bonne introduction aux problèmes de politique économique liés à la propriété intellectuelle. En particulier, le chapitre écrit par Bernard Caillaud, "La propriété intellectuelle sur les logiciels", présente une discussion équilibrée des conséquences de la brevetabilité pour l'industrie du logiciel.

détail des relations de travail entre développeurs libres, ainsi que des stratégies des organisations, fondations et groupes d'usagers qui coordonnent le développement du logiciel libre et comblent les lacunes d'un système fondé uniquement sur le volontariat.

Les références fournies ci-dessous se veulent très sélectives, et peuvent servir de début à une exploration plus large du sujet.

Références

[1] Anderson R. (2002) : "Security in Open versus Closed Systems – The Dance of Bolzmann, Coase and Moore", Cambridge Working Paper.

[2] Baake, P. et W. Wichman (2003) : "Open Source software, competition and potential entry", manuscript.

[3] Benkler Y. (2001) : "Coase's penguin, or, Linux and the Nature of the Firm", *Yale Law Journal*, **112(3)**.

[4] Fuggetta A. (2003) : "Open source software–an evaluation", *Journal of Systems and Software*, **66(1)**, pp.77-90.

[5] Gaudeul A. (2003) : "The (La)TeX project: A case study of open source software", Toulouse Working Paper, Août 2003.

[6] Gaudeul A. (2003) : "Open Source Software Development Patterns and License Terms", Toulouse Working Paper, Septembre 2003.

[7] Ghosh, R.A., Glott, R., Krieger, B., Robles, G. (2002) : "Free/Libre and Open Source Software: Survey and Study", Technical report, International Institute of Infonomics and Berlecon Research GmbH, `http://www.infonomics.nl/FLOSS/report/`.

[8] von Hippel E. (2002) : "Open Source Software as horizontal innovation networks - by and for users", MIT Sloan School of Management, WP No. 4366-02.

[9] Johnson, J. (2003) : "Smoothering intrisic motivation : theory with an application to Open Source", manuscript, Cornell University.

[10] Kuan J. (2002) : "Open Source Software as Lead User's Make or Buy Decision: A Study of Open and Closed Source Quality", Stanford Working Paper.

[11] Lerner J. et J. Tirole (2000) : “Some Simple Economics of Open Source”, *Journal of Industrial Economics*, **52**, pp. 197-234.

[12] Lerner J. et J. Tirole (2002) : “The Scope of Open Source Licensing”, Working Paper.

[13] Mockus A., R.T. Fielding et J.D. Herbsleb (2001) : “Two case studies of open source software development: Apache and Mozilla”, Working Paper.

[14] Mockus A., R.T. Fielding et J.D. Herbsleb (2002) : “Delayed Returns to Open Source Participation: An empirical analysis of the Apache HTTP Server Project”, Working Paper.

[15] Mustonen M. (2002) : “Why do firms support the development of substitute copyleft programs?”, Helsinki Working Paper.

[16] Nichols D. and M. Twidale (2003) : “The usability of open-source software”, *First Monday*, **8(1)**.

[17] Perens B. (1999) : “The Open Source Definition”, in “Open Sources: Voices from the Open Source Revolution”, O’Reilly editors.

[18] Peyrache E., J. Crémer et J. Tirole (2002) : “Some reflections on Open Source Software”, *Communications & Stratégies*, **40**, pp. 139-160.

[19] Peyrache E. (2002) : “Career concerns and choosing whether to compete”, ESEM conference 2002.

[20] Scotchmer S. et P. Samuelson (2002) : “The Law and Economics of Reverse Engineering”, *Yale Law Journal*, **111(7)**, pp. 1575-1663.

[21] Schmidt K.M. et M. Schnitzer (2002) : “Public subsidies for open source ? Some economic policy issues of the software market”, CEPR Working Paper.

[22] Tirole, J., C. Henry, M. Trommeter, L. Tubiana et B. Caillaud (2003) : “Propriété intellectuelle”, Rapport du Conseil d’analyse économique (CAE), La Documentation Française.