Université de Reims Champagne–Ardenne

Faculté des Sciences Economiques et de Gestion

Les logiciels libres : un modèle pour un nouveau mode de production non marchand ?

Mémoire de maîtrise d'économie

mention Analyse des Politiques Economiques

Directeur de mémoire : Pr C. Barrère

PERRET Florian

Année 2003/2004

Introduction

« Aujourd'hui, le capitalisme est traversé par une contradiction liée à l'opposition entre deux modes de régulation de l'économie de la connaissance : le premier repose sur un système de savoirs fermés, fondé sur l'appropriation privée des connaissances : c'est ce qui est en train de s'affirmer notamment aux Etats-Unis, avec le renforcement et l'extension du système de propriété intellectuelle. La deuxième alternative est un système de savoirs ouverts, fondé sur le libre accès aux savoirs » (Carlo Vercelone, 2003).

Un débat fort traverse les théories économiques, il n'en s'agit pas moins que de déterminer quelles sont les évolutions contemporaines du capitalisme. Deux programmes de recherche macroéconomiques avancent des thèses diférentes, le premier, le capitalisme patrimonial, insiste sur la financiarisation de l'économie, alors que le deuxième, le capitalisme cognitif, met en avant la montée du savoir et de l'immatériel dans l'économie. Il nous paraît alors intéressant de nous préoccuper des évolutions qui ont lieu à un niveau inférieur, en ciblant un secteur particulier pour y rechercher ces changements. Les logiciels nous ont semblé être un bon exemple des évolutions plus globales en cours sur les biens informationnels. En effet depuis 20 ans, et l'apparition des logiciels libres, deux conceptions de la production de savoir s'opposent. L'une repose sur le savoir fermé et l'autre sur un savoir ouvert.

La tendance à la dématérialisation de l'économie se retrouve au niveau macroéconomique dans certains indicateurs. Ainsi, l'investissement immatériel est passé de 2.6 à 3.7% du PIB entre mi-70 et mi-80 (soit environ +50%), alors que l'investissement matériel diminuait de 17,5% à 15% (OCDE, 1991). Selon J.W. Kendrick (cité par Horn François, 2000), le stock de capital intangible (éducation, recherche développement, formation, santé) sont devenu équivalent au stock de structure et d'équipement vers 1973 et serait dorénavant supérieur. Aux Etats-Unis, les entreprises du savoirs (communication, éducation, médias, machines et services informationnels) représente 49% du PNB en 1980 (Fritz Machlup, 1984, cité par Horn François 2000). Et selon Jonscher (1984, cité par Horn François, 2000), aux Etats-Unis, la contribution nette des secteurs informationnels (banque, édition, base de donnée) à la valeur des biens des secteurs de la production vendus aux consommateurs finaux s'élève à 42,7% en 1983.
Ces tendances sont concomitantes avec l'apparition de nouveaux modes de gestion, le knowledge management, avec l'augmentation des besoins d'échanges créée par l'externalisation des activités et

l'apparition de la firme réseau.
Beaucoup d'appellations sont utilisées pour décrire ces changements structurels, l'OCDE parle d'économie fondée sur le savoir, d'autres de nouvelle économie ou encore de capitalisme cognitif.

Carlo Vercellone nous donne cette définition du capitalisme cognitif : « un système d'accumulation associant un mode de production capitaliste, un régime d'accumulation privilégiant la connaissance et la créativité, et un mode de régulation caractérisé par des rapports sociaux fondamentaux et des comportements tournés vers l'innovation, la nouveauté et le partage des droits y afférent. Ce système implique une transformation majeure du rapport salarial et des formes de la concurrence ».
Le point de départ de l'analyse du capitalisme cognitif est le constat de l'importance de la connaissance dans la société contemporaine. Elle devient un enjeu de première importance dans la concurrence inter-firmes, où obtenir la primauté d'une information devient stratégique et peut engager des montants financiers importants. Plus globalement, l'objectif des entreprises subi une transformation par rapport à la période fordiste, où la finalité était d'obtenir une production industrielle de masse standardisée. Les perspectives d'accumulation se situe dorénavant plutôt dans des biens dans lesquels le salarié incorpore dans l'objet fabriqué une partie de son intellect. La richesse à acquérir pour l'entreprise, qui lui fournira un avantage comparatif sur ses concurrents, devient alors le cerveau, la pensée même du salarié. Ce nouveau processus d'accumulation nécessite la mise en place de rapports de production nouveaux, antinomique à ceux du fordisme, où le salarié n'était qu'un exécutant effectuant des actions robotisées dans lesquelles toute autonomie, toute réflexion lui est interdites pour ne pas perdre le rythme de la chaîne. Le capitalisme cognitif doit développer de nouveaux modes de travail pour permettre aux salariés d'utiliser ses capacités créatrices et pour pouvoir se les approprier. Le travail est alors mythifié comme devant être flexible et cette nouvelle forme s'insère difficilement dans le salariat. La création est difficilement encastrable dans un lieu et un temps donné correspondant au temps de salariat dans l'entreprise. L'entreprise doit aussi développer ou pousser à la mise en place de structures permettant cette appropriation (les "nouvelles enclosures"), nous verrons que cette question sera centrale pour les logiciels. Ce nouveau concept de capitaisme cognitif est donc une analyse globale que nous mobiliserons ici uniquement aux logiciels et aux biens informationnels.

Les logiciels libres connaissent une croissance importante depuis qu'il existe un système complet et fonctionnel (GNU/Linux en 1991) pouvant couvrir un ensemble de besoins importants sans recourir à des logiciels propriétaires. Certains ont même atteint des positions dominantes sur certains marchés (Apache), alors que d'autres sont en voie de l'obtenir (MySQL). Au départ confiné aux secteurs liés aux réseaux, les logiciels libres s'étendent progressivement à tous les domaines et

semblent suivre une croissance irrésistible. Dans ce sens, l'étude du cabinet Forrester Research (2004) auprès des grandes entreprises américaines (140) nous indique que les logiciels libres ne sont plus réservés aux utilisations liées aux réseaux. Ainsi 53% des entreprises interrogées utilisent Linux pour leurs applications critiques. 44% choisissent GNU/Linux quand ils ont le choix de porter une application ancienne sur un nouveau matériel et 33% pour des applications qui ne fonctionnent que sous GNU/Linux. Cette extension des domaines couverts par les logiciels libres a amené les différents acteurs à faire évoluer leurs stratégies.

Nous tacherons donc de déterminer, quel est le mode de production utilisé par les logiciels libres, et quels avantages et inconvénients celui-ci procure-t-il. Pour cela, nous détaillerons très précisément le déroulement du développement d'un logiciel libre (I.I) en première partie. Nous partirons ainsi des analyses existantes pour les confronter aux dernières évolutions que nous observerons. Cette recherche nous amène à distinguer quatre grands types d'organisations pouvant structurer le développement de logiciel, nous effectuerons alors un lien avec les théories les analysant (I.II). Ceci nous permettra dans la deuxième partie, d'en déduire comment chacune des quatre organisations peuvent répondre à la production de logiciels standards ou spécifiques. Pour ensuite, nous intéresser aux points d'opposition hors marché entre les deux modèles libres et propriétaires (II.I) qui pourraient bloquer le développement libre, puis finir (II.II) sur les extensions possibles, soit uniquement par le type de licence, soit aussi par le mode de développement.

Partie 1: Description et évolutions des modes de production de logiciels libres

Introductions :

- ***Introduction à l'économie des logiciels propriétaires et libres***

L'économie des logiciels présente des caractéristiques particulières dues au caractère immatériel des logiciels, qui permet une duplication à l'infini pour un coût très faible voir inexistant. À l'opposé des coûts de développement importants peuvent exister. Dans un cadre marchand, la production du premier logiciel s'effectue donc à un coût fixe très élevé, puis les autres copies ont un coût marginal insignifiant et permettent de récupérer les dépenses effectuées sur la première copie. Le logiciel étant un bien indivisible, le nombre d'utilisateurs ne fait pas varier le coût de production mais permet d'en amortir le coût.

Le caractère immatériel du logiciel permet d'éviter la rareté dans l'échange et l'usure dans l'utilisation. Le logiciel se situe dans un cadre d'économie d'abondance, la théorie orthodoxe qui traite de l'affectation de biens rares, ne peut donc être utilisée de façon adéquate. Ainsi alors que les entreprises recherchent en général la multiplication de biens rares à un coût le plus faible possible, la logique s'inverse ici où l'objectif devient la recherche pour introduire des coûts de copie de biens d'abondance. La rareté se doit donc d'être organisée par des moyens techniques et/ou légaux qui ont un coût (non exclusivité).

La non usure de ce bien informationnel permet de réfuter l'argument classique de Garret Hardin (Tragedy of the Commons) sur la tragédie des communs. Même dans le cas où ce bien serait accessible à tous, il n'y a pas perte de valeur puisque l'utilisation n'altère pas le bien (bien non rival). La question ne porte alors pas sur l'allocation d'une ressource, et la justification de l'appropriation ne peut pas reposer sur cet argument pour ce type de bien. Les logiciels libres constituent même ce que E. Raymond (1997) appelle un « commun inversé ». Plus le nombre d'utilisateurs augmente et plus le logiciel prend de la valeur : « l'herbe repousse plus haut quand elle est broutée ». Ce principe peut être généralisé à tous les biens réseaux ou informationnels, mais nous verrons qu'il est encore plus important dans le cas de processus de création de connaissances, tel que la production de logiciel libre en particulier.

Ces éléments font que la valeur d'un logiciel a des déterminants non standard. Nous verrons que les logiciels relèvent plus de la valeur d'usage que de la valeur d'échange. Le prix n'est alors plus lié aux coûts de production (par exemple, Microsoft dispose de 85,6% de marge brute sur Windows) mais à

la valeur d'utilisation. Celle-ci ne pouvant être due au logiciel seul, Orio Giarini et Walter R. Stahel (1990) parlent d'une valeur d'utilisation d'un système, prenant en compte à la fois le logiciel, le matériel le faisant fonctionner et les compétences de l'utilisateur (cette dernière évoluant dans le temps après l'achat).

Nous nous attacherons dans notre étude à observer et analyser les différents types de logiciels. Les plus importants étant ceux produits en interne par les entreprises pour leur unique besoin. Ceux-ci ne seront jamais vendus et n'ont aucune valeur marchande. Ils sont très dépendants d'un environnement et donc difficiles à copier ou à réutiliser. Ils représenteraient 95% des logiciels (E. Raymond). Viennent ensuite les logiciels pour entreprises puis, pour finir et bien qu'ils soient les plus médiatisés, les logiciels grand public. De plus, 75% des activités des développeurs sont liées à la maintenance (Frédéric Georges Roux, 1991).

L'objectif d'un logiciel est de permettre une gestion d'information non automatisable par des moyens mécaniques ou électroniques. Il se différencie du matériel par la possibilité de modifier ou de corriger ses actions après sa production. Il concerne donc des processus non stabilisés et non généralisables. La tendance va cependant vers une accentuation de l'automatisme au travers de l'augmentation des routines prises en compte directement par le matériel.

Les logiciels peuvent être dans deux états. Un état dit compilé, dans lequel ils sont directement utilisables, mais où il est impossible de comprendre comment il a été construit, ni de le modifier. L'état de code-source, qui constitue une sorte de recette de cuisine du logiciel, permet d'observer le détail de son fonctionnement et de le modifier. À partir du code source, il est possible de fabriquer le programme compilé. Mais, depuis le programme compilé, il n'est pas possible de revenir au code source, de la même façon qu'il n'est pas possible de retrouver une recette à partir d'un gâteau déjà réalisé.

- ***Introduction aux logiciels libres***

Le logiciel libre s'inscrit dans l'héritage d'une conception très ancienne en informatique. Il prolonge le mouvement UNIX des années 1960. À l'époque les ordinateurs, rares et chers, n'étaient possédés que par quelques grandes entreprises et les Universités. C'est donc au sein des Universités que s'est créée la pratique de la coopération en informatique. Il est alors courant de s'échanger des programmes entre chercheurs et d'en modifier le code source.

L'apparition de la micro-informatique grand public occasionna l'apparition de nouvelles entreprises pour répondre aux besoins nouveaux et différents de ce nouveau public. Les logiciels propriétaires apparaissent à ce moment, la célèbre « lettre ouverte aux hobbyistes » (1976) de Bill Gates marque l'apparition de cette époque. Le président de Microsoft y appelle les hobbyistes (ceux utilisant des

logiciels dans un cadre non professionnel) à acheter les logiciels afin qu'il puisse rembourser ses investissements (90% des exemplaires de son logiciel étaient copiés à cette époque). Bill Gates s'y interroge de la sorte : « Il faut bien acheter le matériel, mais le logiciel est quelque chose que l'on partage. Qui se soucie de rémunérer les gens qui ont travaillé pour le produire ? ». À partir de cette époque, la montée du logiciel propriétaire prohibant le partage, est continue. En parallèle, la communauté UNIX tend à disparaître suite aux multiples versions non compatibles que chaque entreprise cherche à développer pour se différencier. Elle y perd ainsi l'avantage des biens réseaux et se retrouve confinée sur des niches particulières.

Ce n'est qu'avec le projet initié par Richard M. Stallman que le partage des logiciels redeviendra possible. Alors chercheur au MIT (Massachusetts Institute of Technology), il cherche à prolonger la période des UNIX et souhaite un redémarrage de cette philosophie. Il lance en 1984 le projet GNU (qui est l'acronyme récursif de « GNU is Not Unix ») afin de développer un système d'exploitation complet, semblable à Unix et qui soit un logiciel libre. Ce système GNU aboutira, suite à l'ajout du noyau Linux par Linus Torvald en 1991, à l'actuel GNU/Linux.

Pour permettre le développement de ce projet, R. Stallman crée une fondation (la Free Software Foundation: FSF) et une nouvelle licence basée sur le copyright, à laquelle il ajoute des spécificités qui permettent de protéger à la fois l'auteur mais aussi l'œuvre qui apparaît maintenant comme un bien commun. Il utilise en France une comparaison avec « liberté, égalité, fraternité » pour expliciter les spécificités de cette licence :

- Liberté d'accès au code source du logiciel pour l'étudier, le modifier et le distribuer.
- Égalité d'accès au code source et absence de discrimination pour un utilisateur. La règle d'accès doit être la même pour tous (le logiciel peut être payant mais le point précédent permet la redistribution gratuite ensuite).
- Fraternité dans la réutilisation du code source. Tout développeur ayant modifié un code doit remettre l'intégralité de ses modifications dans la même licence. Ce procédé, dit de *copyleft*, permet de conserver le code libre et empêche toute appropriation. Les droits sur le logiciel sont ainsi inaliénables.

Le point majeur n'est donc pas une opposition entre système marchand et logiciels libres (ceux-ci pouvant être vendus) mais entre appropriation et libre diffusion.

R. Stallman considère les logiciels libres comme un mode de développement coopératif et communautaire, à la fois créateur de lien social et productif. Ils sont à ce titre un mouvement social portant des valeurs éthiques. Celles-ci représentent la première raison d'être des logiciels libres avant leur efficacité économique. Les logiciels développés ainsi relèvent de l'intérêt général et peuvent bénéficier au même titre que la recherche, de financements publics (interview donnée à Multitudes n°1, février 2000).

I) Conception de logiciels libres avec le modèle de Bazar

« In the 1980s, Toyota started taking away market share from General Motors and Ford and others. The Toyota car seemed to be made in a way that was less expensive but with better quality. How was that possible? Everybody was fixating on the car. A bunch of researchers from MIT spent time in the factories where the car was being made and wrote a book called, "The Machine that Changed the World." They argued that by focusing on the car, people had missed the big picture. The important thing was the way in which Toyota made cars. The machine that changed the world was not a machine at all; it was a way of making things. » (Steven Weber, "The Success of Open Source" (Harvard University Press, April 30, 2004).

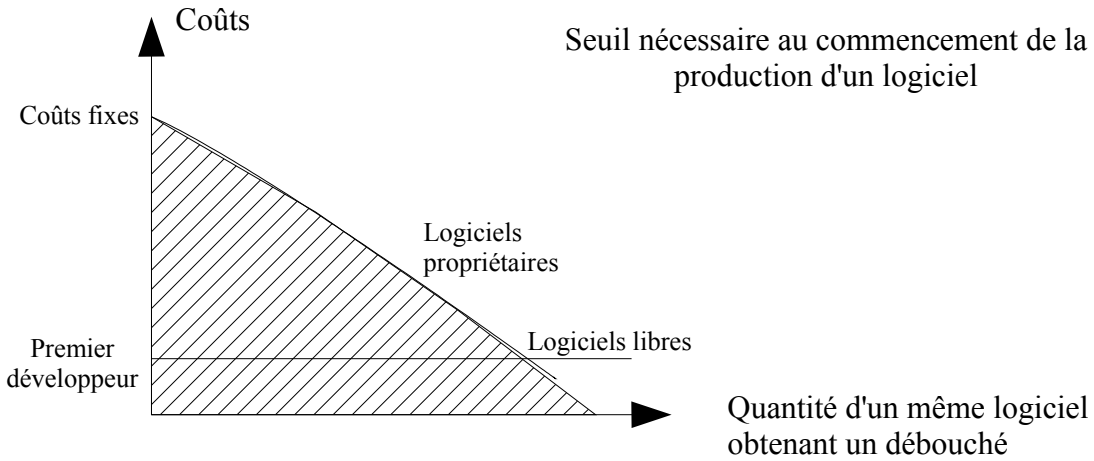
Une des premières analyses du mode de production des logiciels libres est l'œuvre d'Éric Stevens Raymond (ESR) dans son texte « The Cathedral and the Bazaar » (1997). Il y oppose de façon binaire deux modèles de développement des logiciels et propose une analyse du mode de production des logiciels libres en terme de Bazar, ce qui correspond à une structure de réseau qu'il considère être analogue à un marché libre dans lequel les programmeurs décident de participer et sont entièrement libres de leurs actes. Ce système s'oppose alors au mode de développement de type « cathédrale » des logiciels propriétaires, dont l'organisation est centralisée et hiérarchisée.
Dans son analyse du Bazar, E. Raymond établit une étude très fine du développement des projets libres en analysant un projet dont il est l'initiateur. Ces analyses ont très largement été discutées et complétées depuis 1997, et nous intégrerons donc directement certaines de ces considérations. Nous nous attacherons à comprendre l'organisation et les bases nécessaires de ce type de projet. D'autre part, E. Raymond combine à son travail descriptif une interprétation théorique du modèle de développement reposant sur une approche de type libertarienne. Nous discuterons des hypothèses implicites que cette analyse nécessite.

1) Une analyse descriptive du modèle Bazar.

- ***Au commencement d'un projet***

Le choix d'initier un développement libre repose sur un besoin non satisfait. Aucun des logiciels

libres préexistant ne peut accomplir la tâche recherchée et aucun ne peut l'accomplir par simple extension de ses fonctionnalités. Si la personne éprouvant ce besoin est développeur, il peut initier le projet. Dans le cas contraire, un non développeur devra trouver et convaincre un développeur de l'intérêt de la fonctionnalité. Il dispose pour cela de nombreux canaux de communication mis en place notamment dans ce but. Nous retiendrons que l'intérêt d'un premier développeur constitue donc le seuil qui permettra le commencement du projet. Ce niveau se trouve être dans la plupart des cas inférieur au seuil de rentabilité exigé pour le développement d'un logiciel propriétaire (à l'exception des cas où des connaissances spécifiques sont requises, mais nous traiterons de ce cas ultérieurement). Le cas du marché des traitements de textes illustre bien ce point. La suite libre OpenOffice dispose ainsi d'un monopole en langue Kinyarwanda, parlée au Rwanda. Les habitants parlant cette langue étant peu nombreux (et la population connaît peu de langues étrangères) et de plus non solvables pour le prix élevé d'une suite bureautique propriétaire (alors que 90% de la population n'a déjà pas accès à l'électricité). Une communauté d'une vingtaine d'étudiants du Rwanda a décidé de pallier le manque de logiciel dans leur langue maternelle et s'est engagée dans la traduction des 20 000 mots utilisés par OpenOffice. Ainsi, alors que Microsoft Office 2003 n'est disponible que dans 30 langues, OpenOffice dispose de 77 traductions (dont des dialectes indiens, le corse, l'espéranto, ...) qui lui permettent d'être utilisé par des populations non économiquement rentables. Le logiciel libre peut dans ce sens constituer une réponse permettant aux exclus du marché d'utiliser de nouvelles technologies.



L'écart entre les deux seuils (partie hachurée sur le graphique), de rentabilité et du premier développeur, présente une opportunité de niche pour les logiciels libres qui n'ont pas à suivre de critère de rentabilité. Le logiciel propriétaire ne peut que rentabiliser ses coûts fixes élevés en les répartissant sur un grand nombre de clients. Si l'anticipation du nombre de ceux-ci n'est pas assez

importante, le développement n'a pas d'intérêt à débuter dans une logique marchande. Les deux modèles étudiés auront leurs propres limites, pour un logiciel propriétaire la difficulté est de trouver un débouché solvable, alors que pour un logiciel libre elle est de produire le logiciel.

- ***Première diffusion et correction du code source***

Sur le modèle de la pratique de Linus Torvald lors du développement du noyau Linux, E. Raymond suggère de publier le code le plus tôt possible, même si celui-ci demeure très incomplet ou s'il contient un grand nombre d'erreur. Nous remarquerons que le Bazar est donc un mode de développement mais non de naissance d'un projet qui débute dans un cadre plus fermé. Cette méthode de diffusion du code-source, à l'opposé des canons du développement industriel, permet de profiter du principal intérêt d'un développement avec un code source accessible et modifiable par tous. À savoir que tout développeur peut inspecter le code afin d'y repérer les erreurs potentielles. L'avantage du code ouvert repose sur le principe qu'un regard extérieur peut repérer plus facilement les erreurs commises par un autre. Cet avantage s'accroît en même temps que le grand nombre de vérificateurs (effet Delphi), qui dépasse aisément le nombre de ceux qui sont assignés à cette tâche dans une entreprise concevant des logiciels propriétaires.

Selon Lauren Ruth Wiener (1994), « rien n'est plus efficace pour mettre à jour des bogues qu'une utilisation massive pendant des années dans des conditions réelles et impitoyables », « l'usage réel par des utilisateurs réels permet de découvrir plus de bogues que n'importe quel programme de tests ». Ce système est difficile à mettre en œuvre dans un cadre propriétaire avant la sortie commerciale du logiciel. Ce test ne s'effectuera alors qu'ex-post, les premiers utilisateurs peuvent ne pas apprécier un niveau de qualité trop faible, et ils devront attendre une mise à jour postérieure à la sortie, pour disposer d'une version testée et corrigée de cette façon. Dans le cas des logiciels libres, ce type de test s'effectue de façon continue, les versions de développement sont compilées chaque jour et peuvent ainsi être testées sans interrompre le développement.

Pour améliorer l'efficience des recherches d'erreurs, des diffusions au plus tôt et fréquentes sont privilégiées. Ainsi le temps entre la diffusion, le signalement d'une erreur et une nouvelle diffusion corrigée se doit d'être court. Ce système permet d'éviter la duplication des recherches d'erreurs. Le code grâce à cette méthode doit se retrouver plus stable et ceci plus rapidement que dans le cas d'un code propriétaire où l'accès est restreint à une partie réduite des salariés d'une entreprise et où la phase de tests ne peut se faire en condition réelle à grande échelle.

- ***Agrégation de nouveaux développeurs***

Comme nous l'avons vu, le point crucial pour un projet libre est de trouver des développeurs et

notamment d'élargir l'équipe de développement au-delà des personnes initiatrices. Cette étape permet au projet d'atteindre une autre échelle, ce qui lui assure de croître et d'élargir ses possibilités. Ce point constitue une condition pouvant limiter fortement le développement de logiciels libres, il serait l'équivalent de la limite de la rentabilité pour le développement propriétaire. Raymond utilise, pour expliciter le passage réussi de cette étape, le concept de « promesse plausible », que nous supposons emprunté au psychanalyste Jacques Lacan. Ce concept lacanien traduit le lien, souvent implicite, entre le thérapeute et son patient. Ce dernier accepte d'entrer en analyse dans l'espoir que son analyste tienne sa promesse d'aboutir, qui doit lui paraître comme un horizon plausible. Il s'agit à la fois d'une relation de confiance et d'un espoir.

Dans le cas nous concernant, ce concept désigne la capacité des développeurs-initiateurs à rendre crédibles et intéressants à d'autres leurs projets. Les contributeurs étant bénévoles, cette promesse sera la base des motivations de chacun. La promesse est donc la possibilité de mener à terme un projet que chacun jugera individuellement ou collectivement essentiel, et elle sera plausible si les personnes la présentant sont reconnues d'un point de vue technique et relationnel. La réussite dépendant à la fois de capacités techniques et d'innovation, mais aussi de la capacité à intégrer des travaux provenant de personnes diverses. Le jugement porté sur les travaux des nouveaux contributeurs doit paraître pertinent et les décisions doivent être publiques pour permettre une confiance réciproque. Par exemple, le refus d'un correctif doit être motivé, afin de ne pas décourager les contributions.

- ***Quelle taille pour le bazar ?***

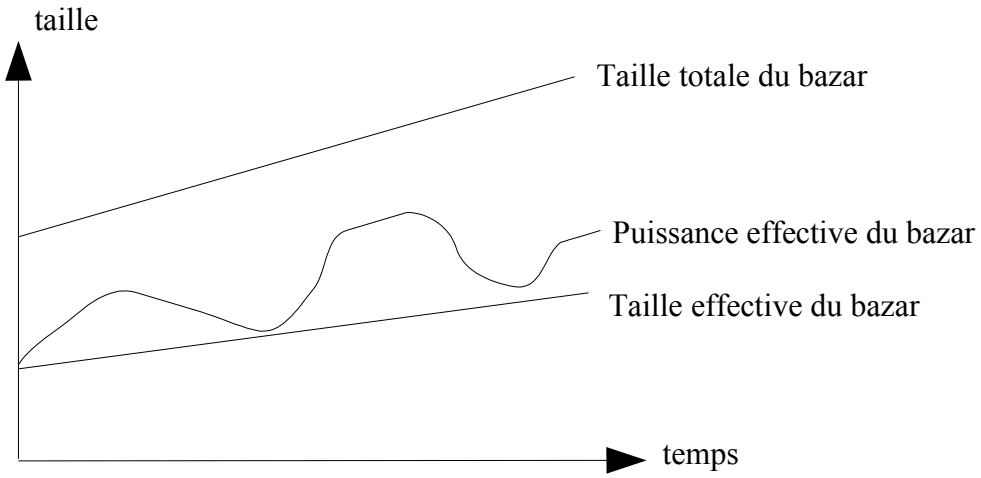
Le développement de logiciel repose quasi-exclusivement sur le travail, l'ordinateur peut être perçu uniquement comme un support complémentaire à celui-ci. Or Frederic P. Brooks (1996) considère que dans le développement logiciel, l'ajout de développeurs supplémentaires sur un projet déjà commencé, ne fait qu'augmenter le temps de développement, effet qu'il appelle « le mythe du mois-homme ». Les tâches de développement ne sont en effet pas divisibles sans demander une augmentation en corollaire des communications entre développeurs, ce qui revient à dire que la productivité marginale du travail décroît. Ce n'est pas le cas d'autres tâches, comme les vendanges, où les communications ne sont pas nécessaires entre travailleurs, et où l'augmentation de la force de travail se retrouvera intégralement dans le résultat. Le raisonnement de F. P. Brooks se base sur l'analyse de l'équipe de près de 1000 développeurs qu'il dirige pour produire des logiciels propriétaires. E. S. Raymond remarque que cet effet semble ne pas fonctionner dans le cas de logiciels libres. Son observation du développement du noyau Linux, pour lequel l'ajout de

développeurs a permis d'améliorer le temps de développement, lui en apporte l'exemple contraire. Il ne propose néanmoins pas d'explication à son observation. Nous essaierons d'en mettre en avant quelques-unes en insistant sur la structure sociale et technique des logiciels libres permettant de diminuer ces coûts de coordination, et donc d'aboutir à ce que les logiciels libres se caractérisent par une productivité marginale du travail croissante.

Nous allons donc reprendre les points ralentissant l'intégration de nouveaux développeurs pour F. P. Brooks. Il s'agit de la formation des nouveaux développeurs au but du projet, à sa stratégie globale, à son plan de travail, ainsi qu'aux technologies utilisées. Dans le cas des logiciels libres, ces étapes sont structurées et ne nécessitent pas d'intervention humaine. Celles-ci ont été mises en place puisque, par essence, les logiciels libres reposent sur l'agrégation de nouveaux développeurs qui peuvent ainsi venir sans avoir à subir une trop grande période entre leur souhait de participer et le moment où ils peuvent le faire, qui pourraient les décourager. Les nouveaux développeurs disposent de *conventions standardisées* entre projets libres. L'organisation, ainsi que les techniques utilisées sont relativement similaires de par l'utilisation de même logiciel de création et de gestion de projet. Ils peuvent de plus trouver sur internet toute une documentation explicative du fonctionnement du projet. Le travail à faire est mis à jour en temps réel et évolue, grâce à un logiciel automatisant ces procédures, en fonction des nouvelles demandes émises par les utilisateurs. Le but du projet et sa stratégie lui sont déjà connus puisqu'il est lui-même utilisateur du logiciel. Et les technologies de développement utilisées sont de plus en plus standardisées par l'utilisation de boîtes à outils, en général GTK ou QT, qui ne nécessitent donc pas d'apprentissage pour quelqu'un ayant déjà participé à un autre projet libre. L'organisation du travail des logiciels libres permet donc de dépasser la limite mise en avant par F. P. Brooks.

Forrest J. Cavalier dans son article « Some Implications of Bazaar Size » prolonge notre raisonnement en recherchant la taille la plus efficace pour le développement en mode Bazar. Alors que Raymond postule qu'un grand nombre de développeurs est nécessaire dès les premières diffusions du code source, F. Cavalier, en partant de l'exemple des débuts du noyau Linux, remplace le critère de la taille par celui du travail effectif. Ceci le conduit à distinguer trois tailles:

- la taille totale du bazar, qui correspond à la taille retenue par E. Raymond. Elle intègre la totalité des contributeurs dans tous les domaines.
- la taille effective du bazar, qui comprend la totalité des participants motivés et dont les capacités individuelles leur permettent d'effectuer une des tâches utiles au projet (modifications, améliorations, documentations, sites internet...).
- La puissance effective du bazar, qui correspond à la taille effective du bazar multipliée par le nombre moyen de contributions en heures par participant.



La taille totale ne peut se suffire pour mesurer le développement, le noyau Linux a ainsi pu démarrer avec un nombre de développeurs réduit mais à la puissance effective très importante. Le nombre de développeurs n'est devenu important que dans un second temps.

• ***Les activités complémentaires au développement***

À contrario de l'approche de E. Raymond, nous noterons qu'une différence trop importante entre le nombre d'utilisateurs et la taille effective peut entraîner des difficultés. Forrest J. Cavalier distingue donc trois types de participants :

- l'utilisateur-demandeur d'aide : ce participant utilise le logiciel car il répond à l'un de ses besoins. Il peut ne pas être programmeur et ne pas être compétent, ou intéressé par la localisation à l'intérieur du code de la portion provoquant une erreur et la corriger. Il peut signaler des problèmes ou demander des nouvelles fonctionnalités.
- l'utilisateur-développeur (ou co-développeur) : ce participant, compétent ou intéressé dans l'avancement du projet, participe au développement, aux commentaires ou aux discussions dans ce but.
- le développeur central (core developer) : ce participant contribue activement dans l'aide ou l'avancement du projet. Il peut travailler sur des portions du code car elles sont nécessaires, même si elles sont difficiles, déplaisantes, ou si elles n'ont que peu d'utilité personnelle. Il sera aussi un contributeur important dans la promotion du logiciel, dans l'aide aux nouveaux utilisateurs...

Si le nombre d'utilisateurs-demandeurs d'aide devient trop important en comparaison des deux autres catégories, ces deux dernières peuvent se retrouver à passer plus de temps à gérer les informations transmises et à répondre aux questions, plutôt qu'à faire progresser le code-source. Ceci pose la question de la répartition des tâches de non programmation ou de leur automatisation.
Un projet se décompose en plusieurs activités spécifiques. Celles liées au développement comme la programmation ou la correction d'erreurs. D'autres y sont complémentaires, comme la réalisation d'une documentation, l'assistance aux nouveaux utilisateurs, ou la maintenance d'un site internet.
Il est nécessaire que chacune de ces activités trouve un nombre suffisant de personnes pour les assurer. Ces activités ne sont pas nécessaires dès le début du développement, mais le deviendront dès que le nombre d'utilisateurs deviendra important. Le développement de ces activités non-informaticiennes permet de plus, d'impliquer des personnes aux profils différents des programmeurs qui n'auraient pu l'être, de par leur non connaissance de programmation, sans ces nouvelles activités.

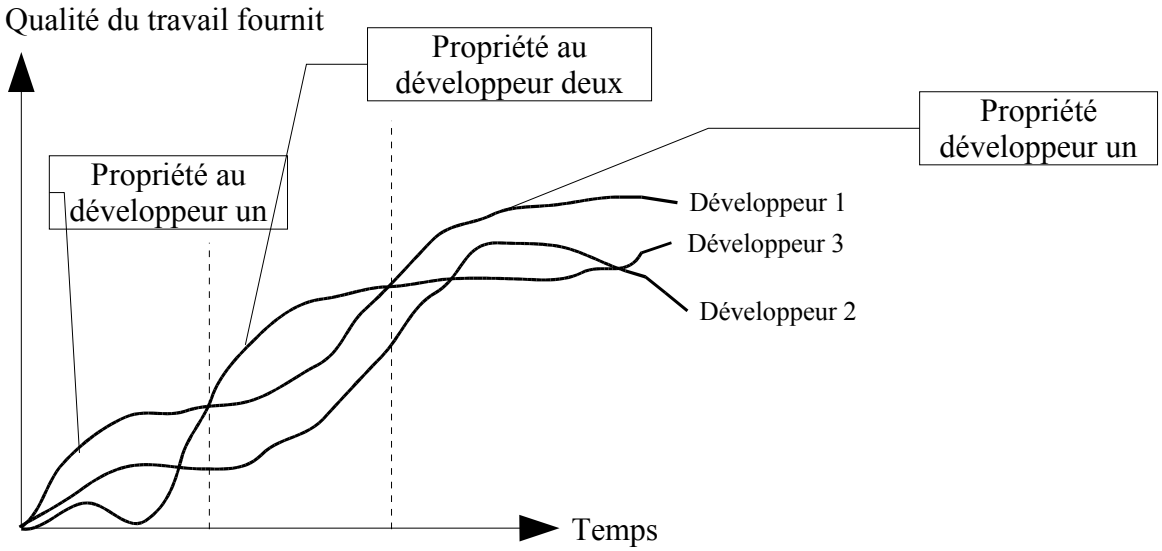
2) Le bazar peut-il être une application de la théorie libertarienne ?

Eric Raymond cherche dans ses articles sur les logiciels libres à développer une analogie entre le modèle dit de « Bazar » et la théorie libertarienne, et cela au prix de quelques contorsions intellectuelles. Nikolaï Bezroukov (1999) parle dans ce sens de Lysenkoïsme, du nom d'un agronome soviétique qui s'opposa à la théorie mendélienne sur l'hérédité génétique et fût écouté car sa théorie se rapprochait de l'idéologie officielle. Nous verrons que pour Eric Raymond, les faits sont têtus et qu'il doit aussi procéder de la sorte pour que les logiciels libres puissent entrer dans le cadre de la théorie libertarienne.
S'il est possible de trouver quelques points communs entre le bazar et les normes du marché parfait, ceux-ci n'en sont pas moins limités. Ces deux approches demandent effectivement une absence de barrière dans l'accès et la diffusion de l'information. Elles reposent aussi sur une concurrence entre projets libres ou propriétaires ayant les mêmes fonctions. Mais le développement des logiciels libres est présenté par E. Raymond comme étant apparu naturellement, et ne reposant que sur des choix individuels non contraints, encadrés par des tabous ou des coutumes. L'organisation, les différentes structures, les motivations individuelles sont le résultat d'un ordre spontané produit par l'Histoire, comme pourrait le qualifier Friedrich Hayek, dans ce sens E. Raymond affirme *« Le verdict de l'histoire semble être que le capitalisme et le libre marché est une façon globalement optimale de coopérer pour engendrer une économie efficace »*. Les logiciels libres sont perçus comme disposant

d'une dynamique menant en tout cas à la réussite du projet. Le développement d'un logiciel libre est ainsi comparé à un chaudron magique (« The Magic Cauldron », E. Raymond, 2000) dans lequel il suffirait de placer les bons ingrédients, E. Raymond fait ici preuve d'un déterminisme technologique absolu. Les logiciels libres auraient donc, en toutes situations, un développement « naturel » vertueux à partir du moment où les développeurs suivent leurs intérêts égoïstes et respectent les tabous. Le résultat en sera de plus, automatiquement supérieur en qualité et inférieur en coût à celui obtenu par un mode de développement propriétaire et ceci quelles que soient la taille et la raison d'être du projet.

Cependant deux barrières doivent être franchies pour développer ce type d'analyse. Le développement de logiciels libres ne permettant pas d'introduire d'échanges marchands, et les licences de type copyleft interdisant toute appropriation du code, Raymond doit pour justifier l'existence d'un marché qui est le coeur de la théorie libertarienne, trouver un moyen de les réintroduire ou d'y pallier. Il imagine alors des droits de propriété, promus comme étant implicites, ceux-ci ne pouvant être raisonnablement présentés comme concrètement existants. Les programmeurs participent au développement pour améliorer leur réputation auprès de leurs pairs ou de leurs employeurs actuels ou futurs. Le développeur le plus méritant deviendra « naturellement » leader du projet et sera reconnu comme tel par ses pairs. Ceci suppose que les développeurs disposent d'une rationalité parfaite leur permettant de reconnaître techniquement et qualitativement le meilleur parmi des centaines, voir des milliers, de développeurs d'un projet. Les activités d'un projet étant assez diverses, ils sont capables de juger et de classer sur de nombreux domaines (marketing, traduction, programmation, site internet, documentation) mais aussi de classer des travaux établis dans tous ces domaines sur une même échelle. Ils savent aussi ne pas tomber dans le piège de la surestimation des membres les plus présents dans les discussions au détriment de ceux s'investissant plus dans le développement. À ce sujet, Nicolas Aurray qualifie le développement des logiciels libres « d'impératif de mobilisation permanente » (2004). Il remarque un phénomène de pyramide inversée, ceux qui contribuent le plus dans les discussions sont ceux qui programment le moins. Sous toutes ces hypothèses, le leader « naturel » est alors décrit comme possédant un droit de propriété implicite sur le code source. Ce droit transférable se trouve être affecté de manière optimale par un marché de la réputation représentée par la courbe d'utilité que chaque développeur égoïste cherche à maximiser, nous pouvons nous considérer comme proches du marché des droits de propriété tel que défini dans un autre contexte par Demsetz, Alchian et Chaing (1973). De son côté E. Raymond fait référence à la théorie des droits de propriété foncière de John Locke. Sur ce marché, les objets d'échange n'y sont pas des portions de code mais le droit de propriété implicite sur l'ensemble du code. Celui-ci s'affecte en dynamique à la personne disposant du plus grand mérite. La concurrence s'établit donc entre des individus, comme peut le défendre Hayek. Raymond

ajoute (1997) *« c'est l'effet dominant du désir de compétition que de produire un comportement de coopération ».* Il n'existe aucune institution dans ce modèle parfaitement décentralisé. Le contrat de licence, pourtant initié par Richard Stallman qui se situe loin de cette conception, peut être perçu comme une règle juridique privée telle qu'elle est recherchée par les libertariens pour assurer la pleine efficacité du marché. E. Raymond conclut d'ailleurs «*je suggère des analogies productives avec d'autres systèmes auto-correcteurs par agents égoïstes ».*

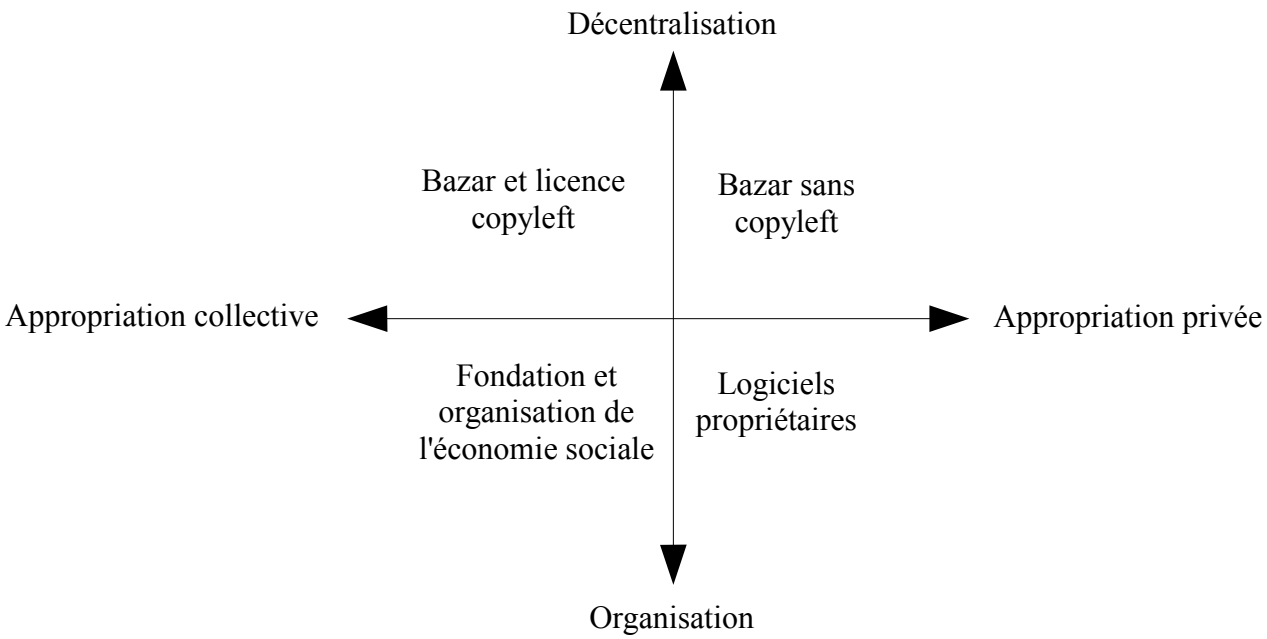


E. Raymond oppose à ce modèle décentralisé et libertarien des logiciels libres, le modèle centralisé des logiciels propriétaires. Si les visions binaires de ce type fonctionnent bien en informatique, elles se révèlent, comme nous allons le voir, relativement faibles dans un cadre d'étude, si l'objectif y est de rendre compte de la diversité des situations rencontrées.

II) Vers une multiplicité des modes de développement de logiciels

Eric Raymond, en ne posant la question des modes de développement de logiciels que d'une façon binaire, ne peut appréhender la multiplicité des modes de production existant. Nous proposerons donc de retenir deux critères pour les discriminer et les classifier. À la fois le caractère centralisé ou décentralisé de la production, critère unique que retenait E. Raymond. Auquel nous ajouterons un critère selon le type d'appropriation du résultat de la production, soit privée soit collective. L'une des

innovations majeures des logiciels libres constitue cette transformation du rapport consommateur et producteur, chacun devenant co-acteur d'un progrès partagé, ce qui inaugure une nouvelle forme des rapports de propriété. Éviter ce critère permet à Raymond de ne pas aborder la question de la propriété des logiciels libres. Interrogation pourtant éminemment stratégique, qui ne lui permettrait pas de se poser en analyseur objectif des logiciels libres et des licences. Cette protection analytique prise, il n'en défend pas moins, et ceci, cette fois, dans un caractère très pratique, un nouveau type de licence permettant une appropriation privée des logiciels libres. Alors qu'au contraire, en introduisant ce critère supplémentaire, nous pouvons comprendre l'intérêt des libertariens à proposer de nouvelles licences et distinguer quatre grands modes de production tels qu'organisés dans le graphique ci-dessous :



Nous posons ces quatre modèles comme étant des idéaux-types. Chaque projet dispose en fait de son mode de développement propre, celui-ci dépendant grandement de relations humaines nouées dans le développement et les personnalités des initiateurs (volonté de déléguer, mode de coordination...). Ces quatre modes de développement définis représentent des généralisations établies après observation de nombreux projets. Ils sont représentatif d'une certaine instabilité du développement des logiciels libres, ainsi l'apparition de fondations constitue une réponse aux manques du développement purement décentralisé et bénévole, mais leur existence est pour le moment trop récente pour déterminer dans quelle mesure elles stabiliseront le développement et les relations entres les différents acteurs (développeurs, salariés, entreprises, Etats).

Le deuxième apport de cette représentation est qu'il est possible d'en faire ressortir que le logiciel libre s'avère être, dans la période contemporaine, la convergence de deux communautés. La première est celle initiée par Richard Stallman, sa cohésion repose sur la défense de valeur éthique comme la liberté et le partage. La deuxième, plus récente, est apparue suite au développement des analyses libertariennes par E. Raymond et a abouti à une nouvelle dénomination : logiciel open-source.

Cette population de développeurs de logiciels libres s'avère donc être duale. Tant leurs motivations diffèrent, cette convergence de deux communautés pourrait paraître relever de la schizophrénie, mais la similitude des objectifs poursuivis lui donne sa cohérence. Ainsi la communauté initiée par Richard Stallman, dont la perspective est un acte émancipateur vers la liberté, a notamment contribué à faire émerger sa contradiction logique qui s'incarne dans les thèses d'Eric Raymond. La communauté représente alors l'intégralité de ces deux sous-communautés présente dans le logiciel libre, œuvrant dans un but unifié sur ce point précis, mais pour des raisons fondamentalement différentes. Cette diversité, certes difficile à gérer et qui génère des conflits internes, permet cependant la progression actuelle des logiciels libres. L'ensemble de la communauté se caractérise donc par son aspect dichotomique et son aspect holistique, que l'on ne saurait dissocier. Les points de convergence principaux de ces deux communautés semblent être la liberté d'expression et l'opposition à Microsoft. Pour le premier point, la conjonction semble complète, il permet la concordance des idées libertaires et libertariennes. Le deuxième connaît par contre des distinctions, l'existence de cet ennemi commun permet pour une grande part d'unifier la communauté, même si les raisons du rejet sont profondément différentes. Pour la communauté défendant la liberté et la coopération, Microsoft est un symbole du logiciel propriétaire, il le représente dans ses caractères les plus orwelliens par son opposition franche aux logiciels libres, par sa défense des brevets logiciels, ainsi que par des pratiques douteuses qu'un acteur si dominant a pu mettre en place dans ses codes source inaccessibles. Pour les libertariens, Microsoft représente la centralisation et le monopole empêchant l'émergence d'un marché parfait. Certains pouvant aller jusqu'à la comparer à une entreprise soviétique. Le logiciel libre devient dans cette optique un moyen de réintroduire la concurrence sur un marché monopolistique. Il n'est donc plus une fin mais un moyen.

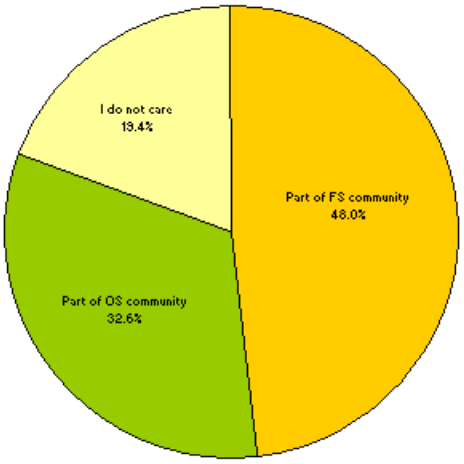
Les entreprises informatiques seront très réceptives à ce message de réintroduction d'une concurrence, puisqu'il leur permet de reprendre une position plus forte face à Microsoft, ou tout simplement de s'ouvrir de nouveaux débouchés. Les entreprises de matériels qui par le passé dominaient les entreprises de logiciels trouvent ici un moyen de prendre leur revanche. Elles peuvent, comme le fait par exemple IBM, se spécialiser dans les services et ainsi proposer des offres intégrées avec matériels, logiciels et services. Les entreprises logiciels actuellement en position

minoritaire, y trouvent aussi un intérêt, puisqu'elles peuvent améliorer leur situation tout en partageant les coûts. Dans ce sens, on peut aisément supposer qu'aucun concurrent à Microsoft Windows n'aurait pu se développer dans un cadre propriétaire. Le coût de développement aurait été très élevé pour une probabilité de réussite très faible (même IBM avec les importants moyens dont il dispose a échoué à imposer son système OS/2). Le logiciel libre permet alors de partager les coûts et les risques. Pour les sociétés de services informatiques (les SSII), le logiciel libre permet d'augmenter les services proposés en ayant la maîtrise du logiciel, alors que dans un cadre propriétaire, seule l'entreprise produisant le logiciel pouvait y apporter des modifications. De fait, dans l'industrie informatique, la totalité des acteurs, à l'exception de ceux qui sont en position monopolistique, ont donc intérêt au développement des logiciels libres. Leur appui actuel, non désintéressé, permet le passage à un rythme de développement plus soutenu et améliore la diffusion ainsi que la promotion des logiciels libres.

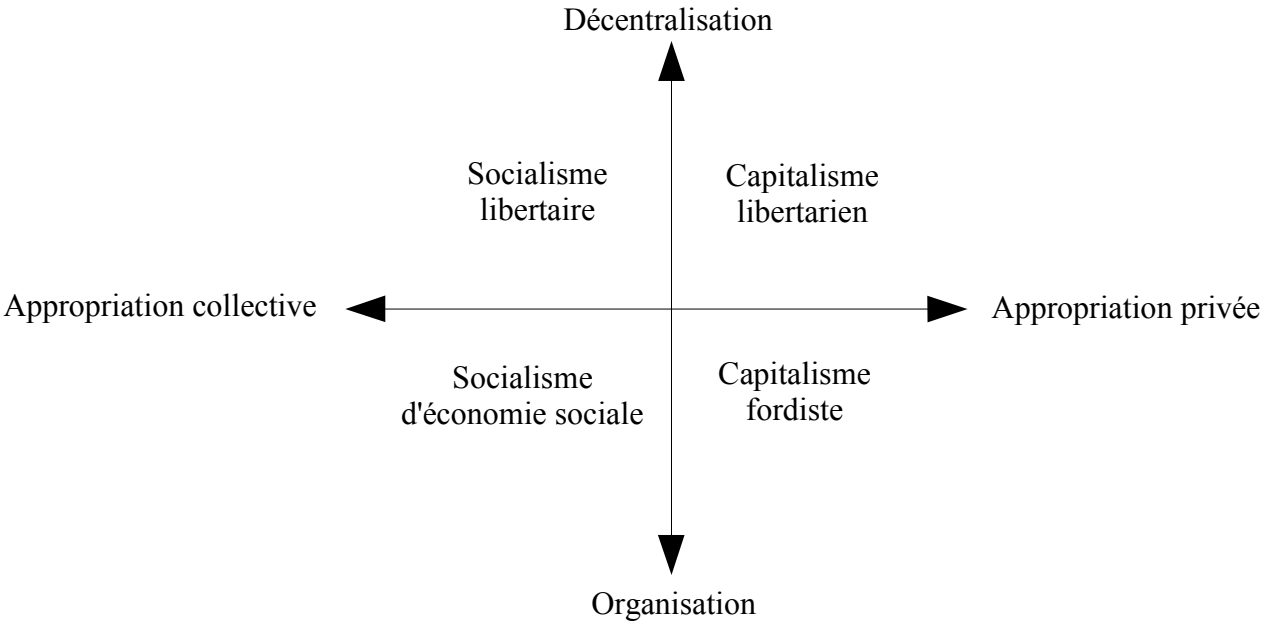
L'existence de ces deux sous-communautés transparaît dans des débats récurrents que nous avons pu observer sur les nombreux lieux d'échanges existants. Par exemple, sur la stratégie à mener, faut-il que les logiciels libres soient un remplacement des logiciels propriétaires ou une simple extension d'un panel de choix s'offrant aux utilisateurs ? La première position serait celle de Richard Stallman, qui ne verrait pas négativement une loi obligeant les administrations publiques à utiliser des logiciels libres et des standards ouverts au nom de l'indépendance technologique des États et du libre accès aux informations publiques. En revanche, les libertariens s'opposent à toute action allant à l'encontre de la primauté du choix individuel et défendent le logiciel libre comme permettant aux utilisateurs d'être plus libres de choisir entre différents logiciels.

La réussite actuelle des logiciels libres repose donc sur la convergence d'une efficience économique et d'objectifs sociaux, ce que nous qualifierons de *processus technico-social*. Une technique élaborée pour un objectif précis et universitaires, permet par la diffusion de son utilisation l'émergence d'un nouveau mouvement social, qui a son tour fait évoluer la technique de base et son environnement juridique afin d'en permettre l'extension.

Le sondage effectué par le Boston Consulting Group et l'OSDN nous indique les proportions des deux communautés. Les partisans des logiciels libres seraient 48% et ceux de l'open-source 32,6%, les autres ne souhaitant pas se positionner.



Le graphique ci-dessous procède à une analogie de nos quatre modes de production avec les grands types de système de production :



1) Le développement en mode bazar avec copyleft

- ***Analyse économique des fondements du logiciel libre***

Les logiciels libres introduisent une rupture dans le mode de production logiciel jusque-là basé sur

un modèle économique industriel ne tenant que peu compte des spécificités des logiciels. Le procédé y était alors de créer de la valeur dans la production comme sur n'importe quel bien industriel. Le Bazar, au contraire, s'est construit sur le caractère immatériel des logiciels. Cette propriété permet d'introduire un mode de développement réticulaire reposant sur l'ubiquité. Des développeurs situés en tout point géographique peuvent travailler au même moment sur des portions similaires du code source.

Les logiciels libres se basent sur les propriétés d'une économie de services. Par la licence empêchant l'appropriation, la valeur d'échange intrinsèque est nulle puisqu'il est toujours possible de récupérer le logiciel gratuitement. En revanche, les logiciels possèdent une grande valeur d'usage qui est la cause du commencement de la production d'un logiciel libre (comme nous l'avons vu précédemment). La valeur d'usage incorporée dans le logiciel peut cependant être étendue par l'ajout de services complémentaires au logiciel. Une entreprise (ou une fondation) peut proposer à un client un mode d'emploi, un service d'assistance technique, voir une adaptation du programme à un besoin spécifique. Cet ajout permet d'introduire un prix, celui-ci n'est pas déterminé fondamentalement sur la valeur du logiciel mais sur la valeur d'usage cumulée par le logiciel et le service associé. C'est pourquoi nous considérons le logiciel libre comme un service dont le prix se cristallise dans l'échange par rapport à cette valeur d'usage cumulée. Il faut noter que ce prix semble relativement rigide, le logiciel étant un bien d'expérience, la formation du prix repose, non pas sur un marché parfait, mais sur la qualité reconnue à l'entreprise. Pour cela l'entreprise a intérêt à embaucher des personnes dont les compétences sont reconnues sur le logiciel concerné. Les développeurs les plus importants d'un logiciel libre trouvent ainsi des emplois leur permettant de continuer à travailler sur ce même logiciel tout en étant salariés pour y apporter des modifications pour certains clients. Le logiciel libre engendre un passage d'une « économie de patrimoine à une économie de l'intelligence » selon l'expression que nous a confié un responsable d'Adelux (partenaire d'IBM). La valeur n'est plus dans le code appartenant à l'entreprise mais dans les compétences de ses salariés, c'est-à-dire dans les modifications qu'ils pourront apporter au code. Le prix ne rémunère alors plus le travail passé de l'entreprise mais son travail futur.

Il faut cependant noter que ce marché des services ne peut apparaître que si le client potentiel n'a pas la capacité de développer ce service en interne sur la base du code source librement disponible. Ce point introduit une deuxième rupture par rapport aux logiciels propriétaires dans lesquels l'entreprise produisant le logiciel dispose d'un monopole sur le service du fait de la restriction d'accès au code source. Un client ne pourra qu'au mieux lui demander une amélioration du logiciel, la décision de la modification ne reposera que sur l'entreprise productrice. Le logiciel libre fait disparaître le marché dans la production de logiciel mais en introduit un nouveau sur les services informatiques.

- ***Analyse du développement***

Le développement des logiciels libres ne peut se définir selon la séparation de l'offreur et du demandeur utilisée généralement en économie. Les concepteurs, les développeurs et les utilisateurs, sont les mêmes personnes, dans le cas de logiciels libres standardisés, nous pouvons dire qu'il y a fusion des fonctions.

De plus, le mode de production réticulaire implique, contrairement au développement linéaire des logiciels propriétaires, une fusion parallèle des trois temps de conception, de réalisation et de consommation. Chacun des progrès de ces trois temps est alors simultané.

Le sociologue Olivier Blondeau (1999) propose d'analyser ce type de développement à l'aide de la métaphore du nœud boroméen. Ce concept, emprunté au psychanalyste Jacques Lacan, représente une notion n'existant qu'à partir de trois. Le nœud boroméen est un ensemble de trois boucles constituées de telle façon que la section d'une de ces boucles entraîne la disparition de l'intégralité du nœud. Le nœud boroméen est donc à la fois une somme de trois parties constituantes et un tout englobant. Dans notre cas, ce tout serait le logiciel libre qui doit pour se développer réaliser l'équilibre de ces trois temps, de ces trois fonctions et de ces espaces. La rupture d'une de ces conditions entraînerait l'arrêt du développement du programme. La promesse plausible que nous avons vue précédemment, constitue alors une raison supérieure que chaque contributeur va observer. Elle donne sens au projet et permettra son évolution dans le temps grâce au maintien de ces grands équilibres.

- ***Analyse de la progression d'un projet : avantages et stratégies pour les logiciels libres***

L'avantage de la pérennisation du développement est accentué par le caractère cumulatif d'un logiciel. Ce point est un avantage certain pour les logiciels libres puisqu'il permet un progrès continu dans le développement sans régressions. Ainsi plus le code déjà développé est important et plus les ajouts postérieurs pourront réutiliser le préexistant et seront donc plus aisés. Nous pouvons considérer qu'il y a sur-additivité dans le développement. Cet avantage, permis par l'accès au code source, est central face au logiciel propriétaire. Ceux-ci bénéficient en revanche d'un temps de développement plus rapide, permis par la mise à disposition, par le salariat, de développeurs à temps plein. L'informatique étant un lieu d'innovation rapide, cet aspect ne doit pas être négligé car il permet de s'adapter aux innovations matérielles plus rapidement. Pour envisager des futurs possibles, il nous faut donc nous interroger sur le point parmi les deux cités précédemment (aspect cumulatif ou innovation) qui sera le plus à même d'assurer un avantage dans le développement.

Cas n°1, l'aspect cumulatif est le plus avantageux. L'apparition de nouveaux logiciels libres devient

de plus en plus aisée par la réutilisation de code préexistant. Cette sur-additivité permet aux logiciels libres de palier un retard dans l'intégration des innovations et ainsi d'être à niveau d'un mode de production basé sur le salariat, voire de le devancer. Le caractère mondial du développement permet aussi de bénéficier d'un avantage dû aux fuseaux horaires. En effet, la répartition des développeurs en tous points de la planète (ou au moins en Amérique et en Europe) permet un développement continu, puisque lorsqu'un continent se repose, un autre peut continuer le travail. Ceci améliore grandement la vitesse de développement par rapport à un type de production géographiquement centralisé.

Cas n°2, le rythme des innovations prédomine. Il impose au développement de logiciels libres une cadence qui ne peut être suivie par la seule contribution de développeurs bénévoles. Les logiciels propriétaires peuvent suivre les innovations en augmentant le nombre de développeurs salariés et distancent ainsi les logiciels libres d'un point de vue technologique.

Cet aspect est un point critique pour le développement futur des logiciels libres. Il semblerait en effet que selon les projets étudiés le cas un ou deux puisse être dominant. Dans le cas d'un désavantage du logiciels libres, nous verrons ultérieurement que les fondations peuvent apporter une réponse.

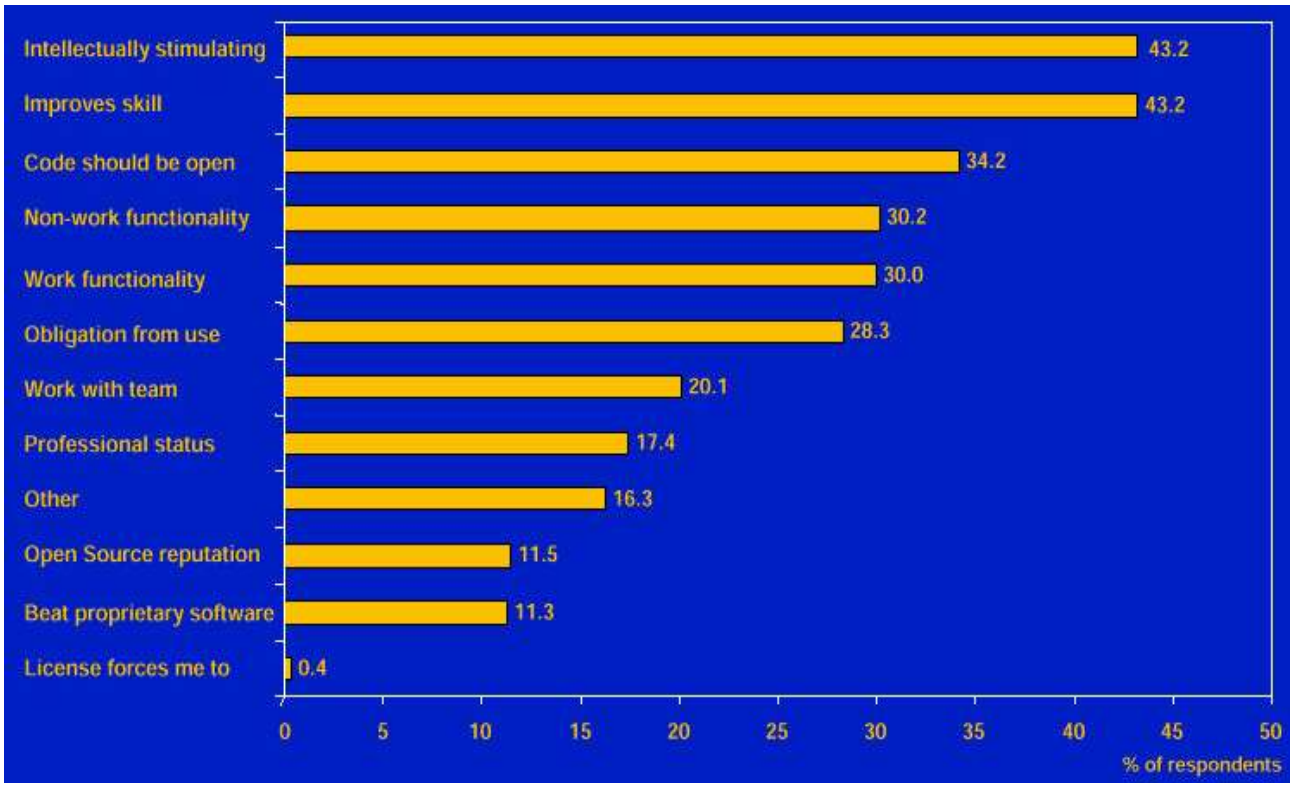
- ***Un modèle proche de la recherche scientifique***

Un logiciel est constitué d'une accumulation de savoirs, François Horn parle d'un « travail intellectuel de production de connaissances codifiées » (2000). La comparaison couramment retenue est celle de la recherche scientifique, dont sont d'ailleurs issus les initiateurs du projet GNU. La logique de coopération en œuvre dans les logiciels libres peut être rapprochée d'un processus de don contre don, tel que peut le définir Mauss. Nous ne retiendrons pas la position du don pur, tel que le défend Derrida, où celui-ci est une rupture avec le modèle économique supposant un retour et un bénéfice. Dans cette conception, le don ne doit pas être la recherche de réciprocité, il existe uniquement de façon univoque. Cette position nous semble trop pure pour que nous puissions la généraliser à l'ensemble des développeurs de logiciels libres, même si elle doit pouvoir s'appliquer pour une partie des individus et plus particulièrement les plus anciens d'entre eux. Au contraire, la notion de don contre don se situe comme une position intermédiaire entre la gratuité et l'intérêt. Le don accepte, de façon consciente ou refoulée, de ne pas être totalement séparé de l'intérêt. Le don se réalise car il existe un espoir d'un don en retour, il n'est plus une forme de désintéressement total, mais n'est pas non plus complètement subordonné à un intérêt. En échange du don du logiciel effectué par les développeurs, les utilisateurs sont invités en retour à participer au développement et à l'amélioration du logiciel. Cette relation peut aussi permettre de comprendre pourquoi les

développeurs ont refusé l'introduction d'une logique monétaire lorsque la société Redhat offrait au 3500 développeurs principaux des actions lors de son introduction en bourse. Même si elle le faisait en compensation du profit qu'elle tirait de leur travail. La position de don contre don strictement individuelle devient sociale grâce aux licences de logiciels libres qui lui créent un cadre ainsi qu'un objectif. Le don y est promu de façon constructive, il crée le lien social et une nouvelle forme d'organisation.

En science comme dans la programmation, la motivation première n'est donc pas monétaire. Une des motivations fortes est la participation à un groupe social constitué qui œuvre dans un même but et dont la motivation est la création d'une œuvre utile à la société. L'UNESCO étudie dans ce sens, la possibilité de classer les logiciels libres comme patrimoine de l'humanité. Mais si cette mobilisation sur des valeurs collectives donne un sens à la somme de travaux individuels, ceux-ci peuvent se comprendre à ce niveau comme une motivation intrinsèque de participer à un processus créatif. Alors que le travail peut être une contrainte, la création est un épanouissement individuel et un moyen d'expression. La motivation de la création est d'autant plus forte qu'elle repose sur un engagement volontaire, la passion créatrice peut alors être une motivation plus motrice que l'intérêt qui ne relève que de motivations extrinsèques (Theresa Amabile, professeur associé de psychologie à l'Université Brandeis, 1986). Linus Torvald a d'ailleurs intitulé le livre où il relate son histoire et celle du lancement du noyau Linux, « Just for fun : The Story of an Accidental Revolutionary » (2001), ce qui exprime des motivations intrinsèques. Un individu peut s'engager dans ce processus pour lui-même sans en rechercher directement de contrepartie. Cette approche est celle de nombreux artistes, et peut être rapprochée dans notre sujet d'étude au développement, qui est une manipulation de symboles formant un langage pour aboutir à une œuvre fonctionnelle.

Une étude menée par le Boston Consulting Group (Lakhani, Wolf & Bates, 2002) étudie les motivations des programmeurs inscrits à SourceForge.Net (le plus important site de projet libre, 837 650 développeurs pour 80 164 projets en avril 2004). Elle nous permet de mieux cerner les développeurs de logiciels libres.



Motivation des hackers, Boston Consulting Group, janvier 2002

Le Boston Consulting Group tire de son étude les constatations suivantes : « l'étude révèle que les [développeurs] recherchent d'abord la stimulation intellectuelle (créativité) et l'amélioration de leurs capacités. Par contre, peu contribuent par opposition aux logiciels propriétaires.
Les [développeurs] peuvent être décomposés en quatre catégories :

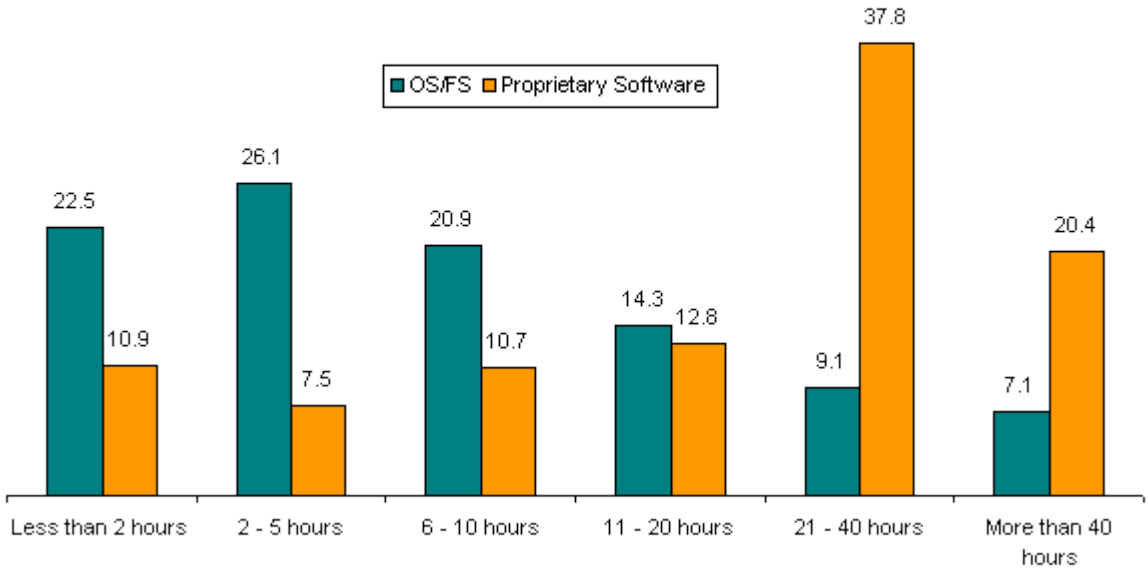
- Les croyants, poussés par la conviction que le code devrait être ouvert (33%),
- Les professionnels, conduits par les besoins du métier et le statut professionnel (21%),
- Les hédonistes, conduits par des besoins privés et la recherche de stimulation intellectuelle (25%),
- Les experts, recherchant l'amélioration de leurs capacités (21%).

L'étude révèle également une forte identification à la communauté. »

Les deux premières réponses nous permettent de rapprocher le développement de logiciels de la recherche. Ainsi la stimulation intellectuelle et l'amélioration de ses compétences, peuvent être similaires à la recherche. Nous pouvons aussi constater que la réputation mise en avant par E. Raymond n'est choisie que par 11,5% des développeurs sondés.
En revanche, une part importante de ces réponses permet de rapprocher le développement des logiciels libres du champ académique, qui repose sur la reconnaissance, l'ouverture et la citation à l'opposé du monde industriel fonctionnant sur le secret. Nous pouvons aussi trouver des valeurs

partagées dans ces deux champs d'activité comme la stimulation intellectuelle, l'amélioration de ses compétences, ainsi que la citation. Le mode de travail présente aussi des similitudes, la vérification et la validation par les pairs se retrouvent dans ces deux types de travaux. Il semble donc pertinent pour les travaux de création intellectuelle.

Nous constatons aussi, grâce à l'étude du FLOSS menée en 2002 auprès de 2784 développeurs, que ceux-ci exercent aussi bien à la fois dans le développement libre que dans le développement propriétaire.



Nous en déduirons donc que le rapport social dans le développement des logiciels libres n'est en définitive pas si éloigné de celui des logiciels propriétaires. En effet, les motivations de la création, comme le développement de soi ou le dépassement de ses capacités, peuvent se retrouver dans le développement de logiciels propriétaires. Les programmeurs informatiques dans leurs activités salariées ne sont pas dans un simple rapport marchand au travail. On peut expliquer le surinvestissement de ces salariés en temps de travail par la recherche de leur développement personnel dans l'objectif d'atteindre le mode de valorisation interne au champ (Bourdieu), c'est-à-dire la création. La différence avec le logiciel libre provient alors du fait qu'ils peuvent choisir sur quel logiciel travailler, ce que leurs statuts de salariés ne permettent pas dans l'absolu, on peut considérer qu'ils auront un surcroît de motivation s'ils choisissent le logiciel sur lequel ils travaillent. Ce surtravail, qui permet aux entreprises de tenir des délais de livraisons très courts, est bien sûr encouragé par la valorisation en interne du salarié effectuant le don de travail

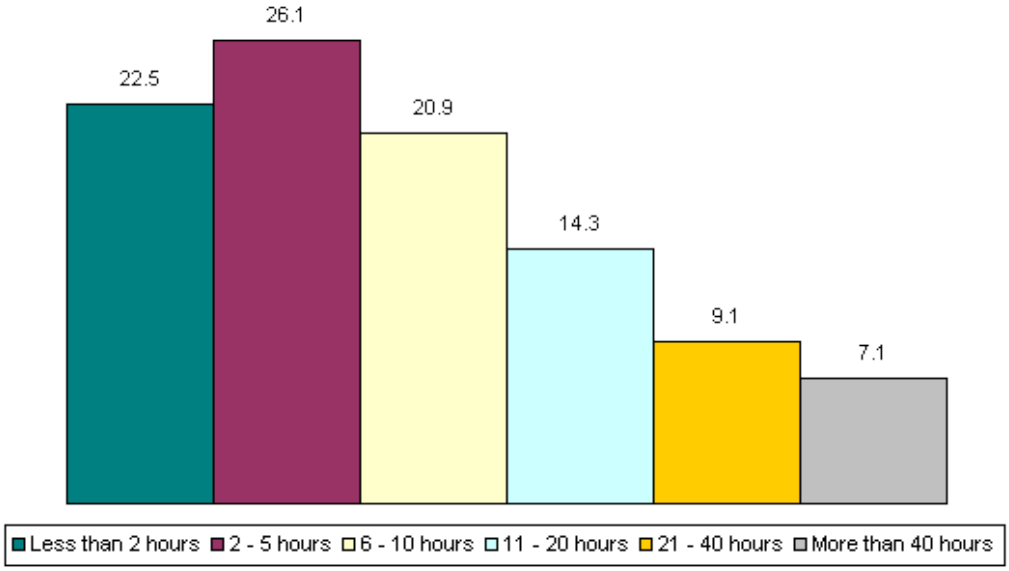
supplémentaire le plus important et donc celui qui aura le plus contribué au code. Une pression du collectif de travail se met aussi en place, le salarié qui souhaiterait rentrer chez lui plus tôt serait mal vu par ses collègues. Cette pression au sein de l'entreprise, entraîne une compétition entre salariés sur la contribution la plus importante faite au logiciel. Mais ce processus qui veut que le don de son surtravail soit la ressource permettant la domination dans le champ de la production de logiciel a un résultat dialectique, puisque qu'il sert aussi à valoriser le travail effectué dans les logiciels libres. Le développeur peut donc programmer la journée comme salarié, et prolonger son activité créatrice en dehors du travail sur un logiciel libre, ceci tout en gardant la même logique de don et de création. Cette similitude est toutefois plus favorable aux logiciels libres, la contribution d'un développeur sera d'autant plus reconnue que son code pourra être diffusé à grande échelle. L'appropriation du code par l'entreprise implique que celui-ci ne sortira pas de ses murs, la reconnaissance du travail ne pourra alors n'être qu'au mieux interne à l'entreprise. En revanche, une diffusion mondiale par internet d'un code ouvert permet une reconnaissance à une toute autre échelle. L'appropriation du code développé peut alors être vécue comme une violence symbolique l'empêchant de valoriser son travail, le logiciel libre lui permettra au contraire de profiter librement de son travail grâce à sa large diffusion.

- ***Profil des développeurs de logiciels libres***

L'étude de FLOSS (2002) nous permet aussi de déterminer un profil des développeurs de logiciels libres. Ils sont âgés de 18 à 35 ans, de sexe masculin à 98,9%, vivent seuls à tout moment ou pour partie du temps à 60%, et sont diplômés de l'enseignement supérieur à 70% (dont 9% de Docteurs). Leurs métiers sont liés à l'informatique à 83,8%, ils sont principalement ingénieurs logiciels (33,3%), étudiants (20,9%), programmeurs (11,2%), consultant informatique (9,8%) ou universitaires (9,8%). Ils habitent surtout en France (15,4%), aux États-Unis (12,7%), en Allemagne (12,6%) ou en Italie (8%). L'Europe héberge plus de développeurs de logiciels libres que les États-Unis, alors que la situation s'inverse pour les logiciels propriétaires. Certains pays du Sud sont aussi présents et voient leurs parts augmenter. L'Inde, qui forme beaucoup d'informaticiens, représente 1,5% et la Turquie 0,3%.

Le temps hebdomadaire que chaque développeur passe au développement de logiciels libres semble important. Il n'est cependant pas possible de discriminer à l'aide de cette étude les cas où ce travail se fait dans le cadre d'un emploi salarié. Nous noterons que la plupart (26,1%) passe de 2 à 5 heures par semaine au développement libre. Les « petits contributeurs » (moins de 5 heures) représentent 48,6%. Les contributeurs « moyens » sont 35,2% (20,9% entre 6 et 10 heures, 14,3% entre 11 et 20 heures). Les contributeurs les plus importants (plus de 21 heures par semaine) représentent la part

non négligeable de 16,2%, dont 7,1% travaillent même plus de 40 heures par semaines.



Nous allons maintenant nous intéresser aux éventuels revenus monétaires apportés par le développement de projets libres. 46,3% n'en ont aucun, 43,3% en ont un indirect et 50,4% en tirent un revenu direct. Parmi ceux en tirant un revenu direct, celui-ci provient majoritairement de la maintenance (18,4%), puis du développement (15,7%), ainsi que 11,9% pour l'assistance technique. Pour les revenus indirects, 17,5% ont obtenu un poste grâce à leurs compétences sur les logiciels libres.

78,9% indiquent cependant que travailler sur un logiciel libre leur apporte de la satisfaction (joyful) (0,4% pour les logiciels propriétaires) et 75,1% attribuent aux logiciels propriétaires une pression sur les délais (1% pour les logiciels libres).

- ***Quelles organisations de la production ?***

Le modèle du Bazar tel que décrit par E. Raymond comme une forme complètement décentralisée n'existe pas de façon pure. Chaque projet dispose en fait d'une organisation propre que nous allons étudier.

Les petits projets peuvent se limiter à un dirigeant, mais quand le projet grandit, il devient nécessaire de partager le travail de supervision, voire de décision. Une personne seule ne peut assumer sur son temps libre l'intégralité de ces tâches. Elle devra déléguer une partie de celles-ci, l'initiateur fixe les modalités de délégation.

Le principe du « dictateur bienveillant », selon la terminologie de Linus Torvald, est le plus

répandu. Il consiste en ce que le ou les initiateurs gardent une certaine autorité sur le code. Il décide souvent seul de l'intégration ou non d'une contribution extérieure et fixe les orientations générales et stratégiques. Cette approche est celle retenue par Linus Torvald pour le développement du noyau Linux. En cas d'absence du responsable, l'affectation du contrôle ne s'effectue pas par un marché, le nouveau responsable est désigné par l'initiateur. Si le responsable s'appelle « dictateur bienveillant », il se doit cependant de respecter un certain consensus. Les licences libres permettent en effet à toute personne de reprendre le code source et d'en distribuer sa propre version. Si le « dictateur bienveillant » ne porte pas de promesse plausible ou s'il refuse injustement des contributions extérieures, il risque de voir les autres développeurs créer ou se tourner vers une autre version (c'est notamment le cas du gestionnaire graphique Xfree qui est remplacé par X.org). Son pouvoir est donc encadré indirectement par la licence, et par le besoin perpétuel de l'aide d'autres personnes, puisqu'il ne peut mener à bien seul le projet.

La recherche du consensus peut d'ailleurs amener le responsable du projet à initier des discussions sur des lieux virtuels publics afin d'obtenir l'avis d'un maximum de personnes impliquées dans le développement. La décision devient alors plus collégiale mais reste en cas de désaccord faite par le « dictateur bienveillant ». Nous verrons, dans la partie sur les fondations, qu'un fonctionnement démocratique peut aussi exister.

- ***Une production encadrée par la technique***

Alors qu'à l'origine des logiciels libres, le développement et les échanges devaient se faire « à la main », la plupart des tâches non créatives du développement sont maintenant automatisées. Auparavant une personne souhaitant apporter une modification devait récupérer le code-source d'origine sur son ordinateur et le décompresser avant de pouvoir apporter une modification. Ensuite, après avoir testé la modification apportée, il déposait la nouvelle version sur le serveur (internet ou avant son apparition BBS). Les modifications ne pouvaient alors avoir lieu qu'une par une et de nombreuses discussions devaient avoir lieu entre développeurs pour accorder toutes les modifications ensemble. Depuis sont apparus de nombreux logiciels centralisés (CVS, Subversion) ou décentralisés (Arch), permettant d'automatiser le traitement des améliorations faites sur le logiciel. Le code source est stocké sur internet, le logiciel utilisé propose une interface permettant de parcourir le code et d'y apporter des modifications. Le numéro de version sera automatiquement mis à jour à chaque changement, le logiciel CVS va alors compiler avec la modification et des tests seront automatiquement effectués afin de valider l'absence d'erreurs importantes dans le correctif. Si les tests sont passés avec succès, la modification est acceptée, sinon l'auteur reçoit les erreurs et est invité à corriger sa modification. De plus, chaque jour le CVS crée une compilation afin que tous les

testeurs puissent la récupérer (plus de 10 000 téléchargements des versions de tests par jour pour la fondation Mozilla) et signaler les erreurs qu'ils remarqueront pendant l'utilisation du logiciel qu'ils feront en condition réelle.

La gestion des erreurs rapportées (la gestion des erreurs inclut la demande de nouvelles fonctionnalités, ceci par souci de ne pas dupliquer les structures à gérer) est aussi, mais plus récemment, automatisée. Le développement par la qualité des logiciels libres est le résultat de cette relation de type feedback avec les utilisateurs. Deux méthodes sont utilisées, la première, utilisée depuis 1997, consiste à placer dans le programme lui-même, une fonction de rapport d'erreurs automatique. Les utilisateurs, même néophytes, peuvent en deux cliques envoyer le rapport. Ce système reste cependant limité aux plus importantes erreurs. Microsoft a d'ailleurs copié ce système pour l'inclure dans la dernière version de Windows en 2002. La deuxième méthode nécessite une démarche de l'utilisateur. Il doit se rendre sur le site prévu à cet effet, et y créer un rapport d'erreurs en décrivant le problème rencontré. Les rapports d'erreurs sont ensuite automatiquement classés et chaque nouvelle erreur se voit attribuer à un « mainteneur » chargé de la suivre. Ce système permet, selon la fondation Mozilla, d'économiser le travail d'une centaine de salariés à temps plein et de traiter entre 75 et 100 nouveaux rapports par jour. Les développeurs participant à leurs corrections sont plus de 1000 au niveau mondial. Ensuite environ 80 personnes sont chargées de valider les corrections. Puis, le dernier niveau de vérification est effectué par 20 « relecteurs » (reviewers). Il s'agit de développeurs clés qui sont parmi les plus prolifiques dans le développement de Mozilla, la fondation leur a donc proposé un rôle plus important. Ce mélange de technique et d'organisation du travail permet de réduire au minimum les tâches pouvant être automatisées et d'utiliser de façon efficiente l'avantage des logiciels libres, d'un grand nombre d'intervenants possibles.

Cependant, les rapports des erreurs s'effectuent exclusivement en anglais, langue de travail des logiciels libres du fait du développement mondial. Ce système peut donc limiter la participation des non-anglophones, ainsi que du public le plus large. De plus, les personnes utilisant ce deuxième médium ne sont pas totalement novices en informatique, le risque existe pour les logiciels grands publics que les attentes des rapporteurs et celles des utilisateurs grands publics différents.

Comme nous venons de le voir, ces gestions automatisées des contributions et des erreurs nécessitent un hébergement sur internet. Ce point s'avère être le seul coût dans la production d'un logiciel libre. Afin de le réduire, des formes de mutualisation ont été mises en place. Le plus gros hébergeur de projet libre est sourceforge.net (plus de 80000 projets en mai 2004), il se propose d'héberger gratuitement tous les projets libres en faisant la demande. Sourceforge.net est une association liée à une entreprise (Open Source Development Network: OSDN, filiale de VA software) proposant des livres et des services de gestion de projet en ligne. L'hébergement des logiciels libres est une vitrine lui permettant d'attirer un nombre important de visiteurs, souvent

qualifiés en informatique, qui peuvent constituer autant de personnes influentes pour obtenir de nouveaux clients. Certaines associations de logiciels libres, comme la FSF de Richard Stallman, proposent aussi gratuitement des services similaires.
Cette mutualisation des seuls coûts fixes existants dans la production de logiciels libres, associée au coût marginal quasi nul, permet à tout projet libre de débuter et de se développer sans coûts, et favorise donc l'émergence des projets.

- ***L'agrégation de logiciels libres : les « distributions »***

Ce qui est communément appelé GNU/Linux n'est pas intrinsèquement un produit. Son état serait plus proche d'un processus dynamique d'agrégation de multiples projets indépendants, dans leurs objectifs et dans leurs rythmes de développement, dont le tout forme GNU/Linux. Les distributions Linux, commerciales ou gratuites, ne sont donc qu'un gel à un moment donné de ce processus. Elles sont une sélection de logiciels, selon des critères propres à chaque distribution et en fonction de ses objectifs (facilité d'utilisation, pour développeurs, utilisation sur des serveurs...). Les concepteurs de distribution ont donc le rôle de produire une solution intégrée avec ces logiciels disparates et ainsi de les rendre cohérents pour l'utilisateur final.
Deux types de distribution peuvent être distingués, les communautaires et les commerciales. Les communautaires ont l'avantage de reposer sur le même système que celui du développement de logiciels libres. Elles peuvent parfois avoir une structure juridique et être démocratiquement organisées. Elles servent d'interface entre les développeurs et les entreprises utilisatrices. Certaines cherchent à obtenir des certifications internationales pour rassurer les utilisateurs sur la qualité de leur distribution. A l'inverse, les distributions commerciales introduisent une rupture. Elles doivent poursuivre un objectif commercial. De nombreuses sociétés se sont créées dans le but de vendre des distributions (RedHat, Mandrake, Suse...), certaines étant même côtées en bourse. L'argent gagné par ces entreprises peut, entre autres, servir à embaucher quelques développeurs pour travailler sur des projets libres dont elles ont besoin, ou il peut aussi servir à la promotion de logiciels libres. Mais pour aider, dans le développement et dans les tests, les développeurs très souvent bénévoles participant à ces distributions, ces entreprises créent des communautés. En retour des contributions, une partie de l'argent gagné sur le logiciel libre y est ainsi réinvestie. Ces compagnies ne sont cependant pas ou peu rentables puisqu'elles ne peuvent utilisées, au contraire du modèle propriétaire, la distribution comme la réalisation du profit où le prix incorpore à la fois le développement et la distribution qui sont effectués par la même entreprise.
Dans le cas du logiciel libre, la distribution en elle-même peut être trouvée sous la même forme sans passer par un rapport marchand. La valeur ajoutée réside donc dans des biens ou services

complémentaires (le manuel, le service après-vente,...). Ces entreprises ont aussi eu le souci de faire sortir les logiciels libres des cercles informaticiens. Elles ont grandement contribué à la simplification de l'installation et de l'utilisation des logiciels libres nécessaires pour toucher un public plus large. En effet, les utilisateurs-développeurs, ainsi que les utilisateurs qui effectuent des retours ou des demandes sur les logiciels, ne peuvent être considérés comme représentatifs du grand public. Les choix pris ne seront pas toujours en correspondance avec des objectifs grand public. De même, la simplification n'a pas toujours été recherchée dans les projets libres. Il a longtemps été considéré que les utilisateurs devaient avoir une démarche d'apprentissage de l'informatique pour utiliser les logiciels libres. Ces entreprises ont été les premières à rechercher par la simplification, l'extension de la base d'utilisateurs, ce qui allait dans leur intérêt financier. Cependant la recherche de la simplification semble aujourd'hui bien acceptée et fait dorénavant partie des objectifs de nombreux projets.

Mais certaines entreprises de logiciels libres, dépassées par des objectifs dialectiques, développent des pratiques dans lesquelles elles cherchent à enfermer le client dans des standards propriétaires tout en lui vendant du logiciel libre. Redhat vient ainsi, après avoir attiré de nombreux clients par de faibles prix, de changer à la hausse sa politique tarifaire (et qui devient un abonnement annuel), tout en la laissant inférieure à des coûts de migration vers une autre solution Linux. Alors que le client était venu vers le logiciel libre pour ne plus dépendre d'une seule entreprise, ainsi que pour l'ouverture, la flexibilité et les personnalisations possibles, l'entreprise fabriquant la distribution recrée un produit fermé qui permet un profit plus facile. Dans le même sens, Suse a longtemps conservé son interface simplifiée d'installation sous une licence propriétaire, se différenciant ainsi de ses concurrents. Maintenant que toutes les distributions disposent d'interfaces d'installation simplifiée, Suse a rendu libre son logiciel (mai 2004).

- ***Faiblesses du développement en mode Bazar***

Le développement de type Bazar dispose cependant de faiblesses dans son mode de fonctionnement. La plus importante est la nécessité de trouver un individu intéressé par un projet pour l'initier. Il n'est pas toujours évident de trouver une personne bénévole prête à y passer du temps. Plus particulièrement, il sera par exemple aisé de trouver des programmeurs, mais il devient déjà plus difficile de trouver des graphistes pour réaliser les images nécessaires aux logiciels. Bien que cette profession, pourtant une des plus informatisée, pratique occasionnellement la programmation. Les projets n'ayant que des besoins mineurs en graphisme peuvent subvenir à leurs besoins, mais ceux dont le projet est par essence graphique, comme les jeux vidéo, éprouvent plus de difficultés,

l'engagement demandé aux graphistes y étant plus important. De même, il s'avère encore plus difficile de trouver des professions moins habituées à programmer, telles que des grammairiens. Ceux-ci disposent de connaissances spécifiques nécessaires à l'élaboration d'un correcteur grammatical. De fait, à ce jour, aucun projet de ce type n'a pu émerger, d'autant plus qu'il est nécessaire de disposer de grammairiens dans chacune des langues, le développement en Bazar perd ici, de par la diversité linguistique, l'intérêt de son caractère mondial.

2) Le développement en mode bazar sans copyleft

- ***L'open-source comme medium de transformation des logiciels libres en logiciels propriétaires marchands***

Open-source est un terme proposé par Eric Raymond, ethnologue, programmeur et ancien collaborateur de Stallman. Il désigne des logiciels sous licence non copyleft. La première apparition de ce terme en 1998 s'inscrit dans le cadre du texte « Goodbye, free software ; hello, open source ». Cette publication fait suite au choix de Netscape de libérer le code de son navigateur internet après avoir lu « The Cathedral and the Bazaar » (Raymond, 1997).

Open-source défend alors une approche des logiciels libres plaçant sur le premier plan l'intérêt pratique et non les valeurs éthiques. C'est ainsi qu'il cherche à amener des entreprises à libérer leurs codes pour profiter de l'aide gracieuse de communautés. Il incite aussi les entreprises à réutiliser les codes sources publiés sous licences non copyleft pour développer leurs propres logiciels propriétaires et marchands à moindre coût.

Pour mener à bien cette entreprise, il promeut un nouveau type de licence aux conditions plus lâches que celle de la GPL. Ces licences, qu'il appelle open-source, se doivent de respecter certaines conditions fixées. Elles doivent être librement distribuables, le code source doit être disponible à tous, les travaux dérivés doivent être possibles et ceux-ci peuvent choisir une licence différente (y compris propriétaire).

La GPL n'apparaît plus alors que comme une licence open-source particulière, possédant des restrictions additionnelles. Ce qui permet aux libertariens de présenter les licences open-source comme plus « libres » que la GPL, dans le sens où un utilisateur dispose de plus de possibilités. Cependant celles-ci s'effectuent au détriment de l'œuvre, qui peut changer de statut, et de l'auteur qui ne peut plus contrôler le devenir de son projet.

Ces licences se différencient donc majoritairement par la possibilité de rompre le cycle de développement en licence libre. Il est en effet possible pour une entreprise de réutiliser un logiciel open-source pour en faire un produit propriétaire. L'argument avancé pour justifier cette possibilité est que les développeurs ne s'intéressent pas à l'utilisation faite de leurs œuvres, leur seul intérêt serait que les entreprises ayant cette possibilité soient en concurrence parfaite. *« Alors que la plupart des développeurs à sources ouvertes ne sont pas intrinsèquement opposés à l'idée que d'autres profitent des cadeaux qu'eux font à la communauté, la plupart exigent également qu'aucune partie [...] ne se retrouve dans une position privilégiée pour tirer profit d'un projet. T Codeur-Lambda accepte que Toto & compagnie puisse gagner de l'argent en vendant ses logiciels ou ses correctifs, si T. C-L peut, lui aussi potentiellement, en faire autant »* (E. Raymond, 1997). Autrement dit, tous les développeurs connaissent parfaitement le modèle de concurrence parfaite et souhaitent le mettre en œuvre, même au prix d'une perte de contrôle sur leurs créations. Les libertariens réussissent par cette voie d'une nouvelle licence, le tour de force de réintroduire le marché dans un secteur où ils n'étaient pas présents.

3) Le développement structuré au sein d'une fondation

- ***Rôle d'une fondation***

Un développement récent et peu étudié jusqu'alors est l'apparition de structures à but non lucratif pour encadrer le développement de logiciels libres. Celles-ci possèdent majoritairement un statut de fondation californienne à but non lucratif (de type 501c). L'émergence de fondations constitue une nouveauté dans le sens où l'économie sociale était peu présente dans le logiciel jusqu'alors. Nous nous intéresserons plus particulièrement à la fondation GNOME qui prend en charge le développement du logiciel éponyme, qui constitue une des interfaces graphiques majeure du système d'exploitation GNU/Linux. Cette nouvelle forme d'organisation se développe pour les grands projets. Elle se doit d'être structurante mais aussi de ne pas imposer trop de contraintes qui décourageraient le travail bénévole. La création d'une fondation chapeautant un logiciel permet d'extraire celui-ci des contraintes de commercialisation. La fondation opère une césure entre les entreprises commercialisant le logiciel et le Bazar le produisant. En servant d'intermédiaire, la fondation permet au développement de maintenir ses choix propres en dehors de contraintes directes commerciales. Cette nouvelle organisation peut donc pérenniser le développement de logiciels libres, tout en s'adaptant aux nouveaux intérêts qui leur sont portés.

Pour le cas GNOME, la création de la fondation répond à un double objectif. Premièrement, mieux coordonner un nombre devenu plus important de développeurs (leurs effectifs ont plus que doublé en deux ans depuis le commencement du logiciel). Deuxièmement, l'arrivée à une certaine maturité du logiciel a amené des industriels et la presse à s'y intéresser. La fondation peut constituer un interlocuteur unique permettant la discussion avec ces nouveaux arrivants.

L'histoire de GNOME, créée en opposition à une autre interface graphique pour GNU/Linux non entièrement libre (KDE), peut permettre de comprendre les principes adoptés par la fondation. La recherche d'une grande ouverture, d'une transparence totale, ainsi que d'offrir la possibilité à tous de participer aux décisions, permet à la fondation de réaffirmer la promesse plausible qu'elle porte. Elle peut, grâce à cette promesse et aux valeurs affichées, attirer les nombreux développeurs recherchant à développer un élément libre manquant. Les fondations de par la matérialisation du sentiment d'appartenance qu'elles créent, ainsi que par l'extension du champ de la promesse plausible qu'elles portent, pourraient donc être une nouvelle voie de développement prenant de l'importance.

- ***Organisation de la fondation***

Auparavant les décisions stratégiques pouvaient être prises par une seule personne (Miguel de Icaza, l'initiateur de GNOME) avec éventuellement des discussions privées avec quelques personnes. Au contraire, la fondation doit apporter représentativité et transparence dans son fonctionnement.
Pour que la fondation soit acceptée par la totalité des développeurs participant au projet, le choix se porte vers un mode de désignation démocratique. L'ancien système où une personne décidait seule occasionnait des tensions entre développeurs et était peu apprécié des entreprises. Des élections mondiales sont organisées chaque année pour désigner les 11 membres du « board of directors » chargés de prendre toutes les décisions (une décision du board of directors peut être annulée par un référendum demandé par au moins 10% des membres). Pour permettre la transparence toutes les discussions sont publiques et archivées. Le Board sera chargé d'animer et de représenter la fondation. Elle a comme objectifs de faciliter l'intégration de nouveaux développeurs et d'éviter les conflits. Elle intègre aussi les entreprises en leur explicitant un mode de fonctionnement de la fondation et de la communauté auquel elles ne sont pas habituées. Elle finance des rencontres physiques entre développeurs, elle crée des documentations et des outils de communication marketing (presse, entreprise, particulier), elle reçoit les dons et fixe les objectifs et la vision à long terme pour le logiciel.

La fondation peut être aussi perçue comme un moyen d'encadrer la pratique des entreprises souhaitant participer au développement. En effet, les contributions ne sont pas considérées par la fondation comme venant d'une entreprise qu'elle ne reconnaît pas comme contributeur, mais comme provenant d'un individu qui peut éventuellement être salarié d'une entreprise. Cette nuance primordiale permet de s'inscrire dans la continuité du développement en mode Bazar et d'assurer l'indépendance des décisions prises grâce à la conservation du pluralisme. Une entreprise souhaitant participer au développement doit accepter de ne pas peser sur les décisions stratégiques. Si elle souhaite une amélioration dans le logiciel, elle peut demander à l'un de ses développeurs de la créer pour qu'il puisse ensuite la soumettre à la fondation, qui demeure seule juge de l'intérêt d'intégrer un développement additionnel au programme. Il est ainsi courant que des entreprises en concurrence sur la distribution ou les services d'un logiciel, coopèrent dans la production de celui-ci (100 de leurs salariés travaillent ensemble à temps plein sur GNOME). Le fait que le développement provienne uniquement d'individus permet d'éviter de subir des pressions, la décision sera prise selon des critères techniques ou selon les choix stratégiques de la fondation, et non selon l'intérêt particulier d'une entreprise.

De même, le choix des dirigeants lors des élections annuelles s'effectue selon le principe de la méritocratie. Chacun vote pour les candidats déclarés en fonction du mérite qu'il a montré lors de son travail sur GNOME. Pour éviter la prise de contrôle de la fondation par une entreprise, les statuts interdisent que plus de 40% des élus travaillent pour une même entreprise.

Les entreprises intéressées dans le logiciel peuvent cependant siéger dans l'advisory board en payant 10 000$ de droits d'entrée (gratuits pour les organisations à but non lucratif et pour les entreprises de moins de 10 salariés). Dans ce bureau, qui ne peut prendre aucune décision, elles peuvent une fois par an discuter des évolutions du logiciel et obtenir un contact direct avec les développeurs élus. Il ne faut pas non plus négliger qu'elles en tirent une image positive auprès des membres de la communauté, qui sont aussi souvent des clients. Les entreprises adhérant à l'advisory board de GNOME sont parmi les plus importantes en informatique, notamment IBM, HP, Novell, SUN, Mandrake, Redhat,... L'existence de ce lieu d'échange permet de contenir les pressions que pourraient être amenées à faire les entreprises. Il leur donne un cadre où elles peuvent soumettre des idées tout en sachant que les décisions ne leur reviennent pas.

- ***Les changements introduits dans le développement***

Le changement le plus important dans le mode de développement est l'introduction d'un développement temporel. Comme nous l'avons vu précédemment, la plupart des projets libres n'ont pas de critère de temps pour leur sortie, qui n'intervient que quand les fonctionnalités prévues pour

cette version sont prêtes et testées sur une période assez longue. La fondation GNOME, après les déboires connus pour sortir la version 2.0 repoussée de nombreuses fois, prend la décision de passer à un mode de développement temporel. La date de sortie d'une nouvelle version est fixée à tous les six mois. Chaque portion du projet souhaitant être de cette version doit impérativement être disponible et positivement testée sur une longue période pour y être acceptée, ce qui maintient la qualité. Ce changement permet des sorties régulières, et ainsi les industriels peuvent planifier leurs sorties sur le même calendrier. Si l'un de ceux-ci souhaite une fonctionnalité particulière, dont il aurait besoin pour des raisons commerciales à une date précise, libre à lui de demander à des développeurs qu'il salarie, de participer au développement de celle-ci afin de l'accélérer et qu'elle puisse ainsi être intégrée à temps dans la nouvelle version.

Une deuxième modification est la possibilité de définir des orientations au travail. La fondation GNOME peut ainsi organiser des « bugs days », périodes où les développeurs sont invités à se consacrer exclusivement au tri et à la résolution des erreurs. Ces périodes, qui peuvent s'étaler sur plusieurs jours, permettent avant une sortie officielle d'une nouvelle version, d'améliorer le niveau de qualité et de fiabilité. La recherche d'erreurs n'est pas l'activité préférée des développeurs, elle s'avère répétitive et peu valorisée car moins créative. L'existence d'une fondation permet donc de demander aux développeurs d'effectuer ces tâches considérées comme ingrates mais nécessaires à la réussite du projet. Plus généralement, les activités moins créatives mobilisent moins les motivations des bénévoles. Les documentations, autre part du projet faisant peu appel à la création, sont aussi rédigées par la fondation. Celle-ci permet donc de dépasser les limites du projet pour les tâches où la motivation intrinsèque ne peut être la création.

Partie 2 : Deux modèles de production en concurrence

L'analyse des modes de production des logiciels libres et la détermination de quatre modes idéaux-types de développement, nous permettent dans cette seconde partie d'étudier comment chacun de ces modes peut répondre aux deux types de logiciels existants (standards et spécialisés). Nous nous attacherons, comme dans la première partie, à ne pas faire preuve de déterminisme technologique mais à voir quel peut être le panel de réponses que ces modes apportent à un type de production précis, chacun disposant alors de ses avantages et de ses inconvénients propres.
Nous nous intéresserons par la suite aux points de confrontation hors marché entre système libre et propriétaire. Puisque quelles que soient les qualités techniques du système de production, les deux formes de concurrence hors marché que nous étudierons peuvent altérer voir compromettre les qualités, voire l'existence, d'un développement libre.
Dans un troisième temps, nous nous intéresserons aux extensions qui s'inspirent, dans l'esprit ou dans la forme, des logiciels libres. Nous verrons dans quelle mesure, les modes de développement des logiciels libres ou les licences qu'ils utilisent peuvent se diffuser à d'autres secteurs et quelles en sont ou en seraient les conséquences sur l'évolution de ces secteurs.

I) Avantages des modes de développement en fonction des besoins

Nous introduisons à partir de maintenant une différence entre logiciels standards et logiciels spécialisés et nous ne limiterons plus notre étude au développement de logiciels libres standards. La spécialisation des logiciels devient alors un nouveau domaine de concurrence entre logiciels libres et propriétaires. Nous chercherons à prendre en compte l'existence de fondations et d'entreprises qui élargissent, en apportant des différences dans le mode de production, le domaine couvert par les logiciels libres.

Le logiciel standard répond à une logique d'offre. Il est aussi appelé progiciel (contraction de produit et de logiciel), et se doit de répondre à des besoins multiples et variés. La logique économique sera donc de cumuler un nombre important de fonctionnalités afin de pouvoir amortir le coût du développement et ainsi de pouvoir cibler la plus large population d'acheteurs-utilisateurs possible. Le corollaire est que chaque utilisateur n'utilise qu'une infime partie du logiciel, l'étude d'Eurostaf (1996) avance le chiffre d'une utilisation moyenne de seulement 20% des fonctionnalités d'un

progiciel par un utilisateur. Si l'augmentation des possibilités d'un logiciel répond à une logique économique d'abaissement des coûts, la recherche de cette sur-fonctionnalité peut aussi entraîner des ralentissements dans la vitesse du logiciel et des complications dans sa facilité d'utilisation. Il faut donc veiller à ce que les défauts ne dépassent pas les avantages des économies d'échelle. Ceux-ci sont permis, puisque tous les logiciels vendus seront identiques, par les coûts de reproduction qui seront ainsi dans ce cas très faibles. Ce point leur permet de bénéficier de dépenses de développement importantes du fait de leur rentabilité ex-post anticipée.

Dans le cas d'un logiciel spécialisé, les fonctionnalités sont faites sur-mesure et le logiciel sera unique. Le logiciel, alors dans une logique de demande, peut être assimilé à un service. Le client rédige un cahier des charges qui contient les fonctionnalités souhaitées dans le logiciel et le prestataire de service devra tenter de s'y conformer. Nous noterons cependant qu'une forte asymétrie d'informations existe sur le plan technique. La rédaction du cahier des charges se fera ex-ante par échange entre le client et le prestataire. Le premier affine dans cette relation ses besoins, dont l'expression en terme informatique est peu aisée. Le deuxième répond sur la faisabilité technique des demandes et essaie d'amener les clients dans un cadre de réflexion proche de la logique informatique. Le contrat rédigé ainsi sera forcément incomplet de par la difficulté de connaître et d'exprimer les besoins. Puisque l'entreprise s'engage sur un produit futur non réalisé dont les caractéristiques détaillées dans le contrat sont incomplètes, le choix du prestataire s'effectuera donc principalement sur sa réputation et sur ses travaux passés. Ces deux points permettent de réduire l'asymétrie d'informations et donc l'incertitude sur la qualité potentielle de l'entreprise, où plus précisément sur les qualités des connaissances internes à l'entreprise incorporées dans ses salariés. La valeur d'une entreprise devient dans ce cas la connaissance de ses salariés qui, bien que comptabilisée comme passif dans le bilan, devient le principal actif de l'entreprise.

- ***Des difficultés à obtenir des gains de productivité***

L'objectif d'un logiciel est d'améliorer la productivité d'une activité. Cependant la production de logiciel est par elle-même peu sujette au gain de productivité (François Horn, 2000). Elle se caractérise par un faible réemploi de code préexistant, par une faible automatisation malgré un processus de production répétitif, par un environnement de travail individualiste et artisanal, par l'intégration de nouvelles techniques non encore stabilisées, ainsi que par le manque de rigueur des participants au cycle de développement dans la définition des services et des fonctions d'objets à fabriquer. François Horn (2000) utilise ensuite les travaux de William J. Baumol, Sue Anne Batey Blackman et Edward N. Wolf, qui permettent de qualifier les logiciels de secteurs à stagnation

asymptotique, dénomination qu'il utilise pour les secteurs qui n'ont que peu d'amélioration de la productivité et qui disposent d'un état avancé sur le plan technologique ainsi que d'une utilisation de la main d'œuvre intensive et irréductible. Ce point aboutit à ce que la croissance de la productivité soit inférieure à celle de la demande. Une étude américaine, citée par l'OCDE (1991), prolonge les tendances actuelles et abouti à la conclusion qu'en 2040, toute la population des Etats-Unis (hommes, femmes et enfants compris) devrait développer des logiciels, afin de permettre la perpétuation du taux de croissance actuel. Les logiciels libres pourraient dans cette optique constituer une réponse à cette difficulté en augmentant les gains de productivité dans l'économie des logiciels, par la réutilisation de codes pré-existants.

1) Logiciels standards et logiciels spécialisés

Le tableau ci-dessous synthétise de quelle façon les modes de développement précédemment vus peuvent répondre à des demandes de production de logiciels spécialisés ou créer des offres de logiciels standards.

	Offre de logiciel standard	***Demande de logiciel spécialisé***
Logiciel libre ou open-source	Deux types d'offre : - Logiciels produits par un développement de type « bazar », plutôt pour les petits projets - Logiciels produits au sein d'une fondation à but non lucratif pour les grands projets nécessitant des développeurs permanents ou des compétences spécifiques	Deux types d'offre possibles : - Reprise d'un code libre standard sous licence GPL (copyleft), le logiciel spécialisé reste libre et étend les domaines couverts sous licence libre - Reprise d'un code standard sous licence open-source, le logiciel spécialisé peut soit devenir propriétaire soit rester libre
Logiciel propriétaire	Une grande entreprise crée un logiciel en utilisant des codes internes pré-existants (modèle de la firme évolutionniste). Elle doit posséder une bonne gestion de ses codes et de ses connaissances internes (« The Secret of How Microsoft Stays on Top », Marco Iansiti et Alan MacCormack, Harvard Business School, 2002)	- Une petite entreprise crée le logiciel dans son intégralité. Elle peut bénéficier d'une « niche » si aucune grande entreprise n'entre sur ce créneau. Dans les autres cas, elle peut difficilement concurrencer les plus faibles coûts de développement d'une grande firme. - Une grande firme adapte ses codes ou un logiciel spécialisé à la « niche »

- ***Le développement de logiciels standards***

Le développement de logiciels standards est le plus ancien lieu d'opposition existant entre modèle libre et propriétaire. Les logiciels libres sont apparus sur ce segment puisque, le logiciel n'étant pas unique, il permet la relation sociale de production des logiciels libres par la confusion de l'utilisateur et du producteur. Sur ce segment les logiciels libres disposent d'un avantage de productivité permis par la réutilisation ouverte à tous de tout code-source disponible. Selon Carper Jones, « la réutilisabilité offre la plus grande valeur ajoutée et le meilleur retour sur investissement ». Il distingue douze composants pouvant être réutilisés : l'architecture, le cahier des charges, les plans, l'estimation, la conception et les spécifications, l'interface, les données écran et les éléments d'écran, le code-source, les documents d'utilisation, les plans et tests. Ceci permettrait en moyenne d'augmenter de 65% la productivité et de diminuer de 50% les délais et de 85% les défauts. Frederick P. Brooks met en avant trois critères nécessaires pour une réutilisation efficace : une

bonne conception, une excellente documentation et une indépendance entre modules d'un logiciel. Ces critères nécessiteraient alors un triplement des efforts initiaux dans le développement.

- Logiciels libres standards

Pour les logiciels libres, la bonne conception est le résultat de la révision par les pairs, une bonne documentation est obligatoire pour permettre l'agrégation de nouveaux développeurs et l'indépendance des modules est un objectif. Ces différents points permettent d'obtenir des logiciels fonctionnels rapidement et sans devoir refaire un travail déjà effectué par d'autres. Les logiciels libres introduisent de cette manière de nouveaux gains de productivité globaux dans les logiciels.
Le deuxième point est que le segment des logiciels standards rend possible la contribution de nombreux développeurs. Le nombre de contributeurs représente un aspect important dans la réussite des logiciels libres, le travail y étant majoritairement bénévole, l'implication en temps par individu y est moins importante que pour un logiciel propriétaire mais se trouve compensée par le nombre d'intervenants.

- Les libérations de logiciels propriétaires

Les avantages sur les logiciels standards du développement coopératif, lorsqu'il est possible de créer une communauté autour du logiciel, amènent des entreprises à effectuer des passages de logiciels propriétaires vers les logiciels libres. Nous allons nous intéresser aux motivations pouvant guider un tel choix. Pour l'entreprise effectuant cette démarche, ce changement n'est pas anodin. Elle renonce à la rente que lui procurait son logiciel propriétaire, en permettant à tout un chacun de copier à l'identique ce logiciel et de le revendre. Elle se retrouve donc dans une situation d'incertitude et de concurrence plus forte qu'auparavant. Elle ne peut simplement vivre du logiciel, elle se devra donc de développer de nouvelles sources de valeur afin de se différencier des nouveaux entrants. Elle garde cependant un avantage relatif de par sa meilleure connaissance du logiciel. Les entreprises appréciant peu l'incertitude, quelles peuvent être les raisons les poussant à libérer leur logiciel ?
La première raison est l'échec commercial d'un logiciel disposant de qualité intrinsèque. Les risques pris sont dans ce cas nuls, puisque l'exploitation dans un modèle propriétaire s'est déjà avérée non pertinente. Ce fut le cas de la suite bureautique OpenOffice, concurrente de Microsoft Office. Son éditeur Sun Microsystem, se trouvait devant l'alternative d'abandonner le projet qui représentait un coût important ou de le libérer, ce qui permettait de diminuer les coûts de développement en obtenant l'aide des bénévoles du logiciel libre, et constituait aussi une source de médiatisation, sans commune mesure à l'indifférence dont bénéficiait l'ancien logiciel. Sun ne représente d'ailleurs plus que la moitié du travail effectué sur le code à ce jour (par une centaine de développeurs) et reçoit l'aide d'environ 70 000 personnes toutes tâches confondues (développement, marketing, traduction,

documentation...). Cette part encore relativement importante s'explique par le peu de lisibilité du code-source (8,5 millions de lignes) pour une personne extérieure. Une grande partie du travail consiste donc à créer de la documentation sur le fonctionnement du code. Une fois cette étape franchie, le développement devrait s'accélérer et les coûts pour Sun diminuer encore. Il ne faut pas non plus négliger dans les motivations ayant poussé à la libération du code, la volonté des dirigeants de Sun, dont les relations avec Microsoft sont exécrables, de s'attaquer à la principale source de revenu de leur ennemi (SUN est l'acronyme de Stanford University Network, l'entreprise fut créée par des universitaires de Stanford pour développer UNIX, la montée du système d'exploitation de Microsoft constituera un opposant de poids dans les années 1990). Sun a choisi de différencier sa propre version d'OpenOffice appelée StarOffice en lui adjoignant divers logiciels propriétaires (comme un correcteur grammatical) et des services d'assistance. Cette suite bureautique lui est maintenant rentable. Cette suite bureautique connaît un succès grandissant, il vient ainsi d'obtenir quelques gros contrats, le ministère de l'intérieur français l'a choisi pour équiper les 30 000 postes (commissariats et préfectures), la Chine vient de commander 1 million d'exemplaires...

Une deuxième raison peut être le choix de passer d'une entreprise « industrielle » à une entreprise de service. Cette évolution majeure dans l'objectif de la société nécessite un changement de culture au sein de l'entreprise et une adhésion des salariés aux nouvelles valeurs. Sur ce modèle, Matra Divisions vient de libérer son logiciel de conception et de fabrication assistées par ordinateur Open Cascade (rapport du projet RNTL, « nouveaux modèles économiques, nouvelle économie du logiciel »). Elle peut ainsi alléger sa participation au développement du logiciel et se concentrer sur des services, comme proposer une version spécialisée du logiciel. Sa meilleure connaissance du logiciel lui donne un avantage technique et marketing sur ses concurrents.

Une troisième raison est la recherche d'une amélioration de la qualité du logiciel. Nous avons vu que le mode de vérification par les pairs permettait un haut niveau de qualité et de sécurité. EDF a dans ce but libéré son logiciel Code Aster chargé de la mise au point et de la maintenance des installations de production et de transport de l'électricité (rapport du projet RNTL, « nouveaux modèles économiques, nouvelle économie du logiciel »). Ce changement dans le développement permet d'obtenir des retours d'un plus grand nombre d'utilisateurs et ainsi d'améliorer la qualité. De plus, ce logiciel dévolu à une activité critique ne supposant pas d'erreur, a tout intérêt à utiliser un développement avec accès au code-source afin de réduire le coût important de la recherche des dernières erreurs au coût marginal croissant.

Une quatrième raison est la recherche d'extension et de pérennisation d'un standard. IBM, en choisissant de remplacer dans son offre WebSphere, son propre logiciel propriétaire de serveur par le logiciel libre Apache, a contribuer à en faire un standard utilisé par 70% des serveurs (Netcraft, 2004). Cette position n'aurait pu être atteinte dans le cadre antérieur du logiciel et du standard

propriétaire d'IBM, et il n'aurait surtout pu resister à la montée du standard de Microsoft. Ce choix de participer à un standard ouvert a permis à IBM de rester sur un marché et d'y obtenir une place importante.

Une cinquième raison est d'étendre l'utilisation du logiciel pour montrer sa fiabilité et ses capacités. Pouvoir montrer des exemples d'utilisation réelle du logiciel est le meilleur moyen de réduire l'incertitude sur son bon fonctionnement. Des entreprises proposent ainsi des logiciels sous deux, voire trois licences. Nous prendrons le cas de Trolltech, éditeur du logiciel QT permettant aux développeurs de créer facilement des interfaces graphiques. La première licence est la GPL classique pour les logiciels libres, la deuxième est une licence commerciale, dans notre cas la QPL. La première licence oblige toutes les modifications à être remises sous cette même licence (principe de copyleft), les modifications apportées par une personne extérieure ne seront donc pas appropriables. Elles s'appliquent pour des logiciels qui sont et resteront libres. La deuxième licence permet au contraire de créer des logiciels commerciaux propriétaires, et ainsi de ne pas être obligé de publier les modifications apportées. Mais pour cela, il est nécessaire d'acheter cette licence à la société Trolltech. Cette entreprise combine les deux systèmes. Elle bénéficie d'un développement libre peu coûteux, efficace et rapide, et en retour elle permet l'utilisation de son logiciel pour produire des logiciels libres. Mais elle bénéficie aussi, avec la deuxième licence, de revenu monétaire généré par les entreprises souhaitant utiliser QT pour concevoir un logiciel propriétaire. Ce modèle est donc une source de financement pour les logiciels libres mais aussi un point de passage vers les logiciels propriétaires.

- Logiciels propriétaires standards

Pour les logiciels propriétaires standards, le développement s'effectue par des salariés à temps plein, ce qui permet une plus grande rapidité de développement si la firme peut embaucher assez de développeurs. Ce marché est très largement dominé par de grandes entreprises, de par l'existence de standards privés, d'effets de réseau, ainsi que de barrières à l'entrée correspondant à l'investissement minimum nécessaire dû à la non réutilisation possible d'autres codes-source. Contrairement aux petites entreprises, les grandes entreprises disposent d'une quantité de codes très importante en interne. Elles peuvent mettre en place une gestion interne du code et des connaissances. Ce qui leur permet de disposer d'un avantage concurrentiel proportionnel à leur taille. Ce point peut expliquer la tendance au monopole sur les logiciels propriétaires standards. De petites entreprises peuvent cependant exister tant qu'aucune grande n'existe sur une niche. Mais un monopole sur un marché peut de plus être contaminant. La revenu tiré de celui-ci peut servir à financer des activités déficitaires sur d'autres marchés, le temps d'imposer par diverses techniques (prix prédateurs, standards privés, effets réseau) le logiciel sur ce nouveau marché. Ainsi, la marge brute de

Microsoft sur Windows s'élève à 85,6%, ce qui représente 100% des bénéfices annuels de Microsoft (Securities and Exchange Commission (SEC) de Wallstreet). Ce seul logiciel permet donc de financer l'ensemble des autres activités de la firme (logiciels, services, matériels...), en comblant l'intégralité des pertes générées par les autres activités. Pour ces différentes raisons, la tendance pour les logiciels standards propriétaires sera donc à l'émergence de grandes entreprises. L'évolution des prix peut d'ailleurs être interprétée comme l'existence d'une rente. Celle-ci peut provenir de la qualité relative dans l'usage du logiciel par rapport à celle de ses concurrents mais aussi de la « force des rendements croissants d'adoption » (François Horn, 2000). Sa diffusion deviendra sa principale force de par les économies d'échelle et de par l'utilisation de standards privés et de l'effet club qu'ils permettent. Ce point permet d'expliquer les stratégies de prix prédateur mises en place. Un logiciel peut ainsi être vendu par une entreprise à un prix faible ou nul sur la période nécessaire à le rendre dominant, pour ensuite relever le prix une fois les clients captifs du standard. Il y a alors création d'une rente différentielle importante. Le niveau de profitabilité élevé de certains éditeurs peut s'expliquer de la sorte. L'histoire du logiciel a cependant montré que la réussite d'un nouveau concurrent dans la prise d'une part de marché importante entraîne des chutes de prix, voire la disparition de ces rentes différentielles (le cas de WordPerfect). François Horn compare ces rentes à des rentes technologiques, qui apparaissent lorsque les coûts de conception représentent une part importante des coûts totaux.

- ***Le développement de logiciels spécialisés***

Les logiciels spécialisés sont des biens uniques pour un usage spécifique. Ils ne permettent pas de créer une homogénéisation des produits, même dans le cas de logiciel répondant aux mêmes besoins. La substitution entre logiciels s'avère donc impossible.

Pour les logiciels spécialisés, la commande passée par l'entreprise cliente porte sur des exigences de fonctionnalités, et non sur le produit qui est encore à créer. Il s'avère cependant très difficile de rédiger un cahier des charges définissant ce qu'attend l'entreprise cliente du logiciel sur mesure qu'elle souhaite. D'une part, parce que l'évaluation des tâches que devra effectuer le logiciel est ardue. Mais aussi, plus techniquement, parce qu'un logiciel n'est qu'une série d'instructions, qui sera interprétée par un ordinateur ne disposant ni de logique ni de bon sens, ni de possibilité de déduction, tous les cas de figures à traiter doivent y être prédéfinis. Devant une situation inconnue au mieux l'ordinateur ne fera rien et, au pire, les réactions sont imprévisibles. La rédaction du cahier des charges demande donc d'y indiquer les connaissances tacites et implicites mobilisées par les hommes effectuant cette tâche. Les manques inévitables lors de la création et les erreurs, qui ne se

révèlent souvent que dans l'utilisation ordinaire, amèneront le logiciel à évoluer au cours de son utilisation. La deuxième raison est que les développeurs considèrent la « beauté » du code et l'aspect technique, alors qu'un utilisateur ne souhaite qu'un outil. L'échange entre deux protagonistes ne poursuivant pas les mêmes buts est toujours difficile pour aboutir à une conclusion commune. La troisième raison réside dans l'évolution des besoins de l'utilisateur dans le temps. Dans ce sens, Pierre Yves Gomez (1994) considère que la qualité est une construction sociale. Elle est le résultat d'une adéquation dans les échanges entre un utilisateur particulier et le logiciel. Or l'utilisateur d'un logiciel évolue, principalement par effet d'apprentissage. Ainsi s'il pouvait au départ souhaiter un logiciel disposant d'une grande facilité d'utilisation, l'apprentissage aidant, il trouvera par la suite les dispositifs d'aide encombrants, voire le ralentissant dans son travail. Le logiciel devrait donc pouvoir être aussi évolutif après la vente, ce qui explique la montée des contrats de maintenance associés aux créations de logiciels spécialisés.

L'entreprise émettant cette demande se retrouve devant une alternative. Faire elle-même le logiciel, ce choix suppose de disposer de compétences internes capables de le développer. La réutilisation d'un logiciel libre comme base de développement permet des gains de temps et sera privilégiée dans les cas où il y en la possibilité. Sinon l'entreprise peut aussi choisir de faire faire le logiciel libre spécialisé. De grandes ou de petites entreprises peuvent répondre à cette demande. La petite entreprise retrouve dans ce cas une possibilité d'être au même niveau que la grande puisque l'accès à des codes ouverts de logiciels libres lui permet de bénéficier de l'intégralité des routines mises à disposition en licence libre. Elle redevient concurrentielle par rapport à une grande entreprise ayant accumulé une quantité importante de codes en interne. La grande entreprise peut aussi choisir de répondre à ce type de demande, mais son avantage ne sera plus dans le développement du logiciel. Elle peut en revanche y associer des services (adaptation, maintenance, réparation, déploiement...) ou du matériel, ce qu'une petite entreprise ne peut pas toujours proposer.

Pour les logiciels propriétaires, la situation est la même que pour les logiciels standards. La grande firme garde l'avantage dont elle disposait pour les logiciels standards. Celle-ci ayant la possibilité de réutiliser son code-source interne. En revanche, si aucune grande entreprise n'est présente, de petites entreprises peuvent se développer sur ces marchés de niches.
En revanche, l'arrivée des logiciels libres introduit des changements. Ils ne répondent que depuis récemment à ces demandes de logiciels spécialisés. Puisque ce type de logiciel répond principalement à une demande d'entreprise, il ne peut y avoir, en dehors du cas particulier des entreprises informatiques, de fusion de l'offre et de la demande, et le développement de type bazar ne sera donc pas possible. Les réponses des logiciels libres à ces demandes sont de deux types, avec

d'une part, l'approche de sociétés de services et d'autre part la constitution de fondations. Certaines fondations, comme celle gérant le code de Mozilla, se proposent de réaliser directement des travaux de ce type sur leurs propres logiciels pour assurer leur financement. Elles réutilisent alors leur code-source sur lequel elles sont les plus à même d'apporter des modifications. Les entreprises souhaitant un logiciel spécialisé choisissent en premier lieu la fondation correspondante dont les compétences internes ne sont plus à démontrer. Le prix apparaît alors comme un vecteur d'information assez pauvre sur la qualité du bien, au contraire du postulat hayekien. En effet, le caractère de plus en plus stratégique d'un logiciel pour une entreprise amène celle-ci à porter son choix, non selon les prix mais selon la perception des compétences internes qu'elle a de la société de service. De nombreuses sociétés de services, appelées SSLL (société de service de logiciel libre), se sont créées afin de proposer des développements spécialisés sur des logiciels libres. Dans ce sens, certains salariés licenciés par AOL Time Warner lors de l'abandon de Netscape, ont créé une entreprise, « Disruptive innovation », pour proposer l'élaboration de logiciels spécialisés sur la base du code source libre de Netscape. Ils travaillent actuellement pour le journal « Le Monde » et pour l'Université de Technologie de Compiègne qui souhaite intégrer dans ses propres programmes des possibilités offertes par Mozilla/Netscape. De grandes entreprises informatiques comme IBM (la majorité des revenus d'IBM provient des services) proposent aussi ce type de service. Selon la licence initiale du logiciel, le résultat sera soit libre soit propriétaire mais ces sociétés essaient de convaincre leurs clients de maintenir le caractère libre de leur travail, au moins parce que cela permet de les réutiliser dans des travaux futurs mais aussi par conviction. Ces sociétés, utilisant des bases de logiciels libres, ont intérêt à afficher leurs bonnes connaissances et leurs participations aux projets libres utilisés afin d'attirer des clients par leurs compétences. Chaque entreprise pouvant avoir accès au code-source, cet affichage pour se différencier peut prendre plusieurs formes : participer au développement en y affectant des salariés à temps plein, ou embaucher des développeurs reconnus sur ce logiciel. Le financement d'une fondation gérant un logiciel, permet aussi de voir apparaître son nom sur le site de la fondation et donne la possibilité d'y faire référence dans des documents commerciaux. Cette possibilité agit comme un signe envers les clients potentiels.

2) Une concurrence sur des éléments hors marché

La montée croissante des logiciels libres, à la fois sur les logiciels standards et les logiciels spécialisés, amène les entreprises de logiciels propriétaires à rechercher des éléments de

concurrence hors marché. Nous allons dans ce sens, voir les utilisations qui peuvent être faîtes des standards et des brevets logiciels.

- ***Standards***

Les logiciels pour communiquer entre eux utilisent des formats de fichier, qui sont une forme d'encodage des données. Selon la logique poursuivie, ceux-ci peuvent être de deux sortes. Dans un cadre propriétaire, l'entreprise productrice cherche des pouvoirs de marché. La définition d'un standard privé est un des moyens lui permettant de mettre en place un monopole. Cette stratégie s'avère intéressante si sa part de marché est importante. Elle peut alors créer un effet réseau entre ses utilisateurs et favoriser la migration vers son logiciel ou empêcher tout départ. Dans le cas de logiciels dont les documents sont sujets à des échanges multiples et fréquents (par exemple un traitement de texte), ou dans le cas où l'utilisation du logiciel requiert une interopérabilité, l'avantage du logiciel pour chaque utilisateur augmente au fur et à mesure que le nombre d'utilisateurs croît (effet réseau). Pour cela, l'entreprise doit aller contre une position d'intérêt général qui serait que tous les logiciels effectuant des tâches similaires soient compatibles entre eux. Au lieu de cela, l'entreprise fabriquant des logiciels propriétaires privilégie son intérêt financier personnel par rapport à l'intérêt général de ses utilisateurs. Elle effectue alors des recherches pour créer et maintenir des barrières empêchant toute compatibilité avec ses concurrents. L'effet réseau dans les logiciels n'est donc qu'artificiellement créé. L'utilisation de standard ouvert peut les faire disparaître. Des standards ouverts existent pour tout type de données. Ils sont définis par des organismes internationaux à but non lucratif. Le World Wide Web Consortium (W3C) a par exemple cette tâche pour internet. Il regroupe à ce jour 350 organisations privées (informatique, télécom, logiciel, grande entreprise...) et publiques (centre de recherche, Université...). Cet organisme créé à l'initiative de structures publiques tente d'amener les industriels vers une interopérabilité, et ainsi que leur concurrence ne se fasse pas au détriment de l'utilisateur. Il s'avère cependant plus facile de faire adopter ces standards ouverts aux acteurs en position minoritaire sur un marché ou par des projets libres, que par l'acteur ayant une position dominante. Pour internet, le navigateur dominant, Microsoft Internet Explorer (MSIE), qui représente 90% des utilisateurs, ne gère que très partiellement les standards définis par le W3C. Les développeurs de ce logiciel ont même tendance à implémenter de nouvelles fonctions non standards empêchant la compatibilité avec les autres navigateurs internet. Ce qui a amené, pendant une période non négligeable, les développeurs à créer des sites internet de façon telle qu'ils soient compatibles avec MSIE et non avec le standard du W3C. Un logiciel propriétaire dominant présente toujours le risque de devenir

un standard de fait, et ainsi de supplanter le standard ouvert afin d'en tirer un avantage en créant l'effet réseau. Des associations de logiciels libres demandent, pour éviter cet écueil, que le législateur propose une loi rendant obligatoire l'interopérabilité. De manière générale, les logiciels libres recherchent d'ailleurs la plus grande standardisation possible afin de toujours permettre la lecture des documents créées (même si le logiciel ayant servi à créer le document disparaît), et pour faciliter les échanges entre logiciels ayant les mêmes fonctions. Nous remarquerons que la recherche de l'extension d'un standard ouvert va de pair avec l'augmentation du nombre d'utilisateurs du logiciel libre, il constitue donc un intérêt supplémentaire à l'utilisation large du logiciel libre.

- Interopérabilité dans l'échange

Par exemple, sur le marché récent des logiciels de discussion en ligne « chat ». Microsoft a supplanté, rapidement et facilement avec son logiciel MS Messenger, le leader historique et inventeur du concept, ICQ de Mirabilis. Il lui a suffi pour cela d'intégrer son logiciel dans les dernières versions de Windows. Ce procédé de vente liée, très apprécié de Microsoft (mais moins des juges de la concurrence, voir la condamnation sur un procédé similaire par l'Union Européenne), lui permet de profiter de son monopole sur un marché pour en imposer un autre. Mais la migration vers Messenger n'aurait pas été si rapide et si massive s'il y avait eu interopérabilité, autrement dit si les utilisateurs d'ICQ avaient eu la possibilité de discuter avec ceux de Messenger. Mais Microsoft, dans un souci d'éviction de son concurrent, introduisit des barrières à ce niveau (principe d'exclusion), ce qui lui permit de provoquer une migration rapide vers son propre logiciel.

Les développeurs de logiciels libres ont à l'opposé recherché une interopérabilité la plus large possible. Ils ont mis en place des logiciels (GNOMEmetting, Gaim, Exodus, Psi...) compatibles à la fois avec ICQ et MS Messenger mais aussi avec la totalité des autres logiciels de ce type (AIM, Yahoo messenger...). Ce programme avait donc comme caractéristique d'empêcher l'exclusion d'un quelconque réseau et permettait une interopérabilité entre tous ces standards privés concurrents et avec un réseau ouverts libres (Jabber).

- Interopérabilité dans les formats de fichier

Les logiciels libres sont en général dans une position d'outsider. Lors du commencement d'un développement d'un nouveau logiciel pour un nouveau domaine, ils doivent faire face à la préexistence de logiciels propriétaires utilisant des formats de fichiers propriétaires. Le cas des suites bureautiques est particulièrement saisissant. Ce type d'applications, qui sont parmi les plus utilisées à la fois par les entreprises et par les particuliers, nécessite des échanges de fichiers fréquents et en nombre. La question du format d'échange est donc de première importance pour ce type de logiciel. Un logiciel libre souhaitant être sur le même type de fonctionnalité (traitement de

texte, tableur, logiciel de présentation), se devra donc d'assurer la meilleure compatibilité possible avec les formats de fichiers propriétaires dominants. Par définition ces formats propriétaires, dans notre cas les .doc, .xls et .pwt, ne sont absolument pas documentés. Il est donc nécessaire pour les développeurs de deviner le fonctionnement interne du format et de tenter ensuite de l'imiter. Ce procédé appelé "reverse engineering" prend cependant un temps important puisqu'une majorité des développeurs de OpenOffice travaille uniquement sur ce point. De plus, le travail doit être repris à chaque nouvelle version de MS Office. Il est donc constant et ne peut se faire qu'avec un temps de retard sur la sortie du format propriétaire. Cette fonction d'émulation, même si elle s'avère pratique pour l'utilisateur, ne saurait être pour les logiciels libres, qu'une étape temporaire avant la généralisation d'un format standard ouvert. Les avantages de celui-ci doivent être assez intéressants pour que le fabriquant du logiciel propriétaire accepte de l'intégrer dans son programme. Pour les suites bureautiques, la situation semble aller dans ce sens puisque Microsoft dans la version 2003 de sa suite pour les entreprises vient d'incorporer la possibilité d'enregistrer sous un standard ouvert (XML). La raison principale est une évolution des utilisations des fichiers de ce type. Les entreprises souhaitent pouvoir générer automatiquement et de façon flexible des fichiers bureautiques à partir d'autres applications. Par exemple, le contenu d'un formulaire est saisi sur internet et celui-ci se transforme (technique de parsing) en fichier de traitement de texte qu'il devient aisé d'échanger avec d'autres. Pour créer ce type de mécanisme, les entreprises ont besoin d'accéder aux caractéristiques du format de fichier. Microsoft, pour ne pas perdre totalement son effet réseau, ne libère pas le format .doc mais ajoute une nouvelle possibilité d'extension. Il conserve donc une partie de son effet réseau pour le moment.

- ***Brevets***

"Le businessman : Quand tu trouves un diamant qui n'est à personne, il est à toi. Quand tu trouves une île qui n'est à personne, elle est à toi. Quand tu as une idée le premier, tu la fais breveter: elle est à toi. Et moi je possède les étoiles, puisque jamais personne avant moi n'a songé à les posséder.
Le petit prince : Moi, je possède une fleur que j'arrose tous les jours. Je possède trois volcans que je ramone toutes les semaines. Car je ramone aussi celui qui est éteint. On ne sait jamais. C'est utile à mes volcans, et c'est aussi utile à ma fleur, que je les possède. Mais tu n'es pas utile aux étoiles... " (Saint Exupéry, Le Petit Prince, 1943).

La rupture majeure introduite par les logiciels libres concerne le mode de propriété des logiciels. Les dispositions propres de chaque État à cet égard, ainsi que les évolutions législatives auront donc des incidences stratégiques.
En terme économique, Schumpeter pose la question de l'équilibre à atteindre entre protection de l'innovation, la motivation de celle-ci reposant uniquement sur la recherche de profit, et diffusion

des innovations. Les deux écueils à éviter étant d'aboutir à trop de protections, ce qui amènerait l'entreprise à disposer d'un monopole prolongé sur son invention grâce à la possibilité d'interdire l'entrée ex post à ses concurrents sur un marché. Situation impensable à long terme dans une optique libérale, ainsi que pour Schumpeter puisque les grappes d'innovations ne seraient plus possibles. Le deuxième écueil est qu'une protection trop faible permet l'appropriation du résultat de la recherche par les concurrents. L'entreprise innovante est alors découragée dans son investissement ex ante, puisqu'elle subit le coût de la recherche mais ne profite pas seule de son résultat. Sous ces hypothèses de concurrence et de valorisation de la recherche uniquement par le profit, la question de la protection de l'innovation se résume à un arbitrage entre retour sur investissement et maintien de la concurrence. Cette façon de poser la question évacue les approches juridiques et politiques qui ont d'autres fondements. Schumpeter prône alors la recherche du degré optimal de protection, qui s'obtient par simple calcul.

Cette vision a cependant l'intérêt de poser l'existence de deux enjeux pour l'innovation. L'une au niveau microéconomique concerne l'intérêt individuel, et l'autre au niveau macroéconomique repose sur ce que nous qualifierons d'intérêt général (Schumpeter ne considère pas l'existence d'un intérêt général, il parlerait d'un intérêt individuel au maintien d'une concurrence effective au niveau macroéconomique). Pour ce qui est du logiciel, l'intérêt général est d'obtenir un rythme d'innovation rapide tout en conservant une compatibilité. Le développement des logiciels repose sur une accumulation d'ajouts incrémentaux. L'octroi d'un monopole sur une innovation, pour une période donnée, entraîne le risque macroéconomique de bloquer toutes les innovations qui auraient besoin de réutiliser celle-ci pour progresser. Bessen et Maskin (2000, cités par François Horn, 2000) avancent dans ce sens l'idée que, dans le cas « d'interdépendance combinatoire ou séquentielle des innovations, le brevet peut entraîner à moyen terme une baisse de l'innovation ». Le cycle de vie d'un logiciel étant particulièrement court, une protection de quelques années peut aisément bloquer une génération de logiciels. Partha Dasgupta et Joseph Stiglitz (1980, cités par François Horn, 2000), ajoute qu'une protection longue crée des duplications d'innovation préjudiciables au niveau macroéconomique.

En théorie, le brevet, mécanisme de protection de l'invention, est attribuable selon trois critères : la nouveauté, l'inventivité et l'applicabilité industrielle. Les découvertes scientifiques et l'innovation en sont exclues, puisque non appropriables et librement diffusables. Aux États-Unis, où les brevets logiciels sont autorisés depuis peu, les grandes entreprises informatiques se lancent dans des courses au dépôt de brevet. Amazon a breveté de cette manière les bons d'achats virtuels, Microsoft les barres graphiques indiquant une progression et Amazon la possibilité d'acheter en ligne en 1 clique

de souris. Des individus ou des entreprises peuvent utiliser ce système afin d'obtenir des revenus facilement. Un procès a ainsi eu lieu avec Witold Ziarno, qui a porté plainte contre la Croix-Rouge au sujet d'un brevet sur le fait de faire des dons en ligne. Pangia Intellectual Property a attaqué 40 petits magasins en ligne pour violation de son brevet permettant de faire des achats en ligne, 24 ne pouvant supporter le coût du procès ont payé les 30 000 $ demandés, alors que les 16 autres en partageant les frais ont réussi à gagner le procès. IBM seul dispose de plus de 30 000 brevets logiciels aux États-Unis et Microsoft y dépose 10 brevets par jour. La plupart du contenu de ces brevets ne présente aucune invention. Ils sont cependant acceptés par les offices des brevets dont les revenus sont indexés sur le nombre de brevets qu'ils acceptent. En cas de recours devant un tribunal, ils seraient annulés mais en attendant, ils représentent un moyen de dissuasion important.

Le développement des logiciels est un secteur où cohabite de très grandes entreprises multinationales mais aussi beaucoup de PME. Ces dernières ne disposent pas de la même assise trésorière que ces grands groupes. Le coût de dépôt d'un brevet va de 100 000 à 500 000 $, auquel s'ajoutent les coûts pour le défendre devant les tribunaux de l'ordre de 1,6 million de dollars en moyenne. Les individuels, les TPE et les PME s'en retrouvent donc exclus. Elles ne peuvent se lancer sur un marché où un brevet est déposé faute d'avoir les moyens de le contester et il est facile d'utiliser un dépôt de brevet non valide pour obliger ces petites structures à abandonner un développement. François Horn (2000) cite le cas d'une start-up française, contrainte de cesser ses activités aux États-Unis après le dépôt d'une plainte contre elle par une entreprise détentrice d'un brevet. Adobe attaqué par la suite sur ce même brevet, a eu les moyens financiers pour aller au jugement et a obtenu l'annulation du brevet (Jean-Paul Smets-Solanes, Benoît Faucon, 1999, p.142). Plus globalement, le résultat de l'introduction des brevets aux États-Unis représente, pour les entreprises de logiciels, une augmentation de 20% des dépenses juridiques et une baisse de 10 à 15% de la recherche et développement (Marchand Eric, 2004). Les coûts des litiges sur les brevets logiciels sont ainsi passés de 5 millions de dollars par an en 1982 à 4 milliards de dollars en 1998, faisant le bonheur des cabinets juridiques (Andy Grove, co-fondateur d'Intel, 2004). Ce qui s'avère contraire au but recherché. Le parlement européen a récemment débattu de cette question, pour aboutir à la conclusion que les brevets ne devaient pas s'appliquer aux logiciels, à l'exception de quelques cas très encadrés (la commission doit cependant encore valider ce texte). Ce vote favorable a pu être obtenu grâce à la mobilisation des défenseurs des logiciels libres mais aussi la Confédération Européenne des Association de Petites et Moyennes Entreprises (CEA-PME), à l'Association française des producteurs d'œuvres multimédia (Apom), et au Club Informatique des Grandes Entreprises Françaises (CIGREF), regroupant les grandes entreprises non informatiques. Au contraire, les multinationales effectuaient du lobbying en faveur de l'extension du brevetage.

Pour le cas des logiciels libres, la législation sur les brevets représente un danger important. Elle

rendrait impossible la distribution de certains logiciels. Aux États-Unis, les logiciels libres n'ont plus la possibilité de lire un DVD ou d'encoder de la musique au format MP3, ces deux actions étant brevetées. Il n'est donc plus possible de les proposer gratuitement et le coût d'une contestation de ces brevets est trop élevé pour des individus bénévoles ne disposant pas de moyens équivalents aux détenteurs de ces brevets. À l'opposé des brevets, les licences de logiciels libres empêchent l'appropriabilité du logiciel. Ce système, dans un cadre d'innovation fréquente, permet d'obtenir un développement rapide et sans limitations. Ainsi contrairement aux hypothèses de Schumpeter, le cadre n'est pas ici concurrentiel, et l'objectif de l'innovation n'est pas un profit futur. Nous devons donc redéfinir les deux niveaux précédemment retenus. Au niveau microéconomique, les développeurs du libre souhaitent, dans un souci de productivité, pouvoir innover tout en réutilisant un maximum de code préexistant. L'effet de report (spillover) accélère la recherche / développement en permettant l'accès aux recherches des concurrents (1984, modèle de M. Spence, cité par François Horn, 2000). Comme ils ne cherchent pas de rémunération, une protection du profit ne leur est pas nécessaire. Les objectifs poursuivis deviennent donc les mêmes au niveau microéconomique et macroéconomique, soit le libre accès et la libre diffusion. Les logiciels libres peuvent donc utiliser, comme toute œuvre de l'esprit, le droit d'auteur qui présente l'avantage d'être une protection gratuite, immédiate et de s'appliquer à l'intégralité du code. Il protège l'expression de l'idée et non l'idée elle-même. L'auteur conserve ainsi son droit moral et l'intégrité de l'oeuvre est assurée.

- ***Le futur, une combinaison des standards et des brevets peut-il bloquer le développement des logiciels libres ?***

À l'horizon 2006, Microsoft devrait sortir une nouvelle révision majeure de son logiciel Windows. Ce nouveau produit pour le moment connu sous le nom de code « longhorn », disposera de nombreuses nouvelles fonctionnalités. Parmi celles-ci l'apparition d'un nouveau langage de création d'interface graphique (XAML) couplé à de nouveaux outils de développement correspondant (.net), inspiré de ceux développés par la fondation Mozilla (XUL) depuis 1999. Le point de divergence entre ces deux projets est que celui de Microsoft sera disponible uniquement sous MS Windows et le standard sera privé, alors que celui de Mozilla est déjà disponible sous de nombreux systèmes d'exploitation différents et le standard est ouvert. Ce nouveau système sera au cœur du nouveau système d'exploitation et Microsoft tente actuellement de breveter le maximum de points afin d'en limiter la copie (au rythme de dix dépôts de brevets par jour). Cette nouvelle façon de programmer et la gratuité des outils pourraient attirer les développeurs dont les logiciels libres ont besoin pour se développer, nous avons déjà vu que ce point est le plus critique pour le développement libre. Les développeurs de logiciels libres ont donc décidé de développer des outils (Mono) compatibles avec . net et permettant, à contrario de celui-ci, de développer sous plusieurs systèmes d'exploitation

(GNU/Linux, MacOS, Unix, MS Windows...). Cet outil n'est cependant qu'un élément de défense pour permettre de continuer de développer des logiciels libres sous MS Windows (puisqu'il est impossible de développer un logiciel libre sous licence GPL avec un logiciel propriétaire) et il est possible que ce programme enfreigne plusieurs brevets. Mais le point central est que ce langage se veut le remplaçant des langages actuels de création de sites internet et veut ainsi créer un « nouvel internet » plus évolué sur le modèle des recherches faites par la fondation Mozilla. Il permet ainsi d'exécuter des logiciels depuis un site, et non depuis l'ordinateur. En cas d'imposition de ce standard privé, le risque est alors grand qu'à terme seuls les utilisateurs de MS Windows puissent accéder à certains sites internet et aux nouvelles fonctionnalités. Ce standard privé étant de plus protégé par des brevets, il pourrait exclure les logiciels libres de ce nouvel internet. Ce système bloquerait donc toute extension d'un système d'exploitation alternatif. Nous avons dit que Mono ne peut qu'être une solution de défense afin de maintenir la compatibilité, c'est pourquoi une stratégie alternative a été mise en place. Les fondations GNOME et Mozilla se sont rencontrées le 21 avril 2004, afin de créer un système alternatif multiplateforme et ouvert. Le résultat des travaux menés sera ensuite proposé au World Wide Web Consortium (W3C) afin qu'il en fasse un standard officiel sur internet. Ce standard aurait alors la légitimité et pourrait entrer dans une course aux développeurs afin de bénéficier des rendements croissants d'adoption et ainsi de s'imposer comme un standard de fait face au standard privé. Celui-ci ne devra pas seulement être équivalent au standard privé, il devra le surpasser techniquement et qualitativement afin de s'imposer. Selon Miguel Benasayag (« Resister, c'est créer », 2003), toute alternative souhaitant élargir son cercle d'utilisateurs à des non-militants se doit d'être de qualité supérieure au système majoritaire, afin de justifier le passage à l'alternative par ses qualités intrinsèques et non juste par le choix d'être dans l'alternative. L'importance croissante d'internet et des réseaux en général fait de ce futur standard un enjeu majeur.

II) Quelles extensions possibles du modèle libre à d'autres biens informationnels ?

Le mode de développement des logiciels libres dispose, sous certaines conditions, d'avantages pour la production de biens à forte intensité cognitive. Olivier Blondeau parle d'une modification de *« ces rapports, définissant la place de l'individu dans la société, loin d'être fondés sur la compétition prédatrice, s'inscriraient dans une logique de coopération réticulaire de micro-*

initiatives singulières. La communauté composée par les producteurs de signes et les usagers-consommateurs serait alors tout entière mobilisée, non par des critères de rentabilité financière [...], mais bien plutôt par la recherche d'une meilleure adéquation entre les besoins et le produit, c'est-à-dire par la qualité et l'efficacité sociale ». Ce modèle est donc une source d'inspiration pour des expériences sur d'autres domaines que les logiciels. Nous ne recherchons pas ici une extension macroéconomique mais uniquement les cas d'extension existants. Nous pouvons les classer en deux catégories, les projets reprenant ou s'inspirant de licence GPL soit dans une optique open-access soit dans une approche copyleft, ou les projets s'inscrivant dans les mêmes rapports coopératifs à la production. Dans une question qui lui était adressée lors d'un colloque à l'assemblée nationale française (2001) sur ce sujet, Richard Stallman retient trois catégories d'œuvres concernées.

La première concerne les œuvres fonctionnelles comme les logiciels, le matériel éducatif, les recettes, les encyclopédies... Ce type d'œuvre peut se voir appliquer toutes les libertés, y compris celle de redistribuer une version modifiée. Cette approche permet alors d'améliorer la diffusion et l'utilité de l'œuvre pour la société en lui conférant un statut de bien public.

La deuxième catégorie regroupe les œuvres d'opinion comme les livres d'auteurs et les revues scientifiques. Pour préserver la pensée de l'auteur qui est constitutif de l'œuvre, il est déconseillé d'autoriser la modification. En revanche, l'ensemble des autres libertés de libre diffusion peuvent s'y appliquer, l'œuvre de l'auteur reste intègre et bénéficie alors d'une libre diffusion.

La troisième catégorie concerne les œuvres esthétiques comme certaines œuvres d'art. Ces œuvres peuvent se voir appliquer toutes les libertés de diffusion et de modification.

Une quatrième catégorie, non citée par Stallman, concerne les domaines nécessitant un accès à des données (par exemple, le génome humain pour la biologie qui est une œuvre fonctionnelle uniquement si l'accès aux données informationnelles est possible).

Pour mesurer les possibilités d'extension, nous chercherons à vérifier que les conditions pré-requises ainsi que les conditions d'efficience sont ou pourraient être présentes. Nous avons donc choisi de questionner l'existence d'une population capable de participer bénévolement et donc disposant soit d'un revenu extérieur, soit de revenu généré directement par l'activité. La possibilité d'une production ou d'une diffusion à faible coût (souvent immatérielle) et l'absence de coûts fixes trop importants.

1) Les extensions de la licence

- ***revue scientifique***

Les revues scientifiques sont une extension récente, elles s'inscrivent dans la continuité de l'Initiative de Budapest en 2003. Ces nouvelles revues libres permettent le libre accès aux résultats de la recherche. En effet, alors que la recherche est majoritairement financée sur fonds publics, la publication est assurée par des revues majoritairement privées, qui facturent la publication, puis la consultation des résultats des recherches. Le financement public se répète trois fois à chaque niveau et l'auteur doit souvent renoncer à son droit sur son article et le céder à l'éditeur qui est alors le seul décisionnaire des publications et diffusions futures. Les revues prestigieuses étant obligatoires dans tout laboratoire, leur prix est particulièrement inélastique. Certaines de ces revues demandent donc des sommes dépassant les 10 000 € pour un abonnement d'un an. Ce qui empêche toute diffusion des résultats de recherche dans les pays du Sud et alourdit les charges de fonctionnement des laboratoires du Nord. Le collectif de scientifiques Public Library of Science (PLoS) fondé par le Prix Nobel Harold Varmus et les biologistes Patrick Brown et Michael Eisen décide de lancer en mai 2003 une nouvelle revue de biologie (PloS biology), basée sur le principe de libre accès (open-access). Devant le succès et la reconnaissance obtenus par ce premier essai, il lance ensuite en avril 2004, une revue de médecine (PloS medecine). Les articles de ces revues sont placés sous une licence spécifique non copyleft (licence Creative Commons), ils sont librement consultables par internet, ou sur papier pour un abonnement plus faible que dans le cas des revues propriétaires. Le financement s'opère selon un principe de publieur-payeur, le laboratoire dont la publication est acceptée par le comité de lecture paie la somme forfaitaire de 1 200 €. Ce montant permet de financer le comité de lecture, qui doit lire un grand nombre de publications pour au final n'en garder que peu (20% pour PLoS Biology). Les articles retenus financent donc ceux qui se sont vus refusés. Si ce système permet un libre accès, il ne faudrait pas qu'il devienne une limite dans l'accès à la publication de petites structures, notamment du Sud. Des fondations (Welcome Trust britannique ou l'Institut pour une Société Ouverte (OSI)) proposent des bourses spécifiques dans cet objectif mais un système de tarif en proportion du budget du laboratoire serait moins contraignant.

Ce projet repose, comme pour les logiciels libres, sur une communauté forte dont les besoins de communication en son sein sont élevés. La recherche scientifique a aussi un fonctionnement itératif, les recherches du passé servent à la recherche présente, leurs libres diffusions constituent donc un enjeu important. Les textes de recherche sont immatériels donc la diffusion peut s'effectuer aisément et a peu de coût pour la version en ligne. Cependant la rémunération du comité de lecture, dont les membres ne sont pas bénévoles, crée un coût fixe important nécessitant la recherche de recettes.

- ***Art contemporain***

L'art contemporain a été la première expérimentation d'une extension des licences libres. Ce passage s'est effectué d'autant plus facilement qu'il permet la recherche d'une nouvelle dimension créative. En effet, comme pour les logiciels libres, l'auteur en plaçant son œuvre sous une licence libre, permet au public de son œuvre d'entrer dans sa production. Une licence spécifique est créée, la licence art libre, permettant la copie, la diffusion et la modification de l'œuvre. Elle « reformule le droit d'auteur en permettant au public de faire un usage créatif des œuvres d'art » (artlibre.org). Chaque œuvre devient potentiellement un matériau pour une nouvelle création, comme le pratiquait déjà au début du 20ème siècle, les surréalistes qui n'hésitaient pas à réutiliser une œuvre en en modifiant le contexte pour en détourner le sens. Le copyleft permet une nouvelle pratique coopérative de la création. Les œuvres ainsi créées et diffusées sont de tout type : musique, image, art plastique, photographie, littérature, poésie, vidéo, cinéma...

Ces œuvres ne sont pas créées de la même façon que les logiciels libres puisque la technologie ne permet pas encore de modifier aisément à distance de la musique ou des vidéos. Les artistes créant sous ces licences peuvent vendre le premier exemplaire de leur œuvre, par exemple une œuvre plastique, puis en laisser la libre diffusion et modification ensuite. Ils peuvent donc à la fois subvenir à leurs besoins et garantir le copyleft.

- ***Emissions de télévision***

Le mode de diffusion libre d'émissions de télévision répond aux difficultés financières des petites chaînes de télévision locale, aussi appelée télévision citoyenne, très présente aux États-Unis et au Canada. Le coût de production d'un grand nombre d'émissions télévisées est hors de portée pour ces chaînes associatives dont les budgets varient de 0 pour Génération TV d'Évry à 75 000 €. Les placer sous une licence libre (Licence Publique Multimédia: LPM) permet, dans une logique de don contre don, de disposer d'une quantité plus importante d'émissions en permettant l'échange gratuit d'émissions entre chaînes associatives. Placer directement l'œuvre sous cette licence permet d'éviter de signer pour chaque utilisation un contrat de cession des droits et réduit ainsi les coûts de négociation. Pour conserver l'intégralité de l'œuvre, seule la diffusion complète, générique compris, est permise. Pour éviter une récupération par une chaîne commerciale, le choix s'est porté sur une protection non légale, qui est de tout simplement indiquer « ce film est à destination des télévisions de proximité » dans le générique. Des associations, comme Handicap International, mettent aussi leurs films sous cette licence. Ce type de pratique correspond particulièrement bien aux besoins de

petites structures.

- ***Musique***

La dématérialisation par la numérisation de la musique permet une création musicale à faible coût (coûts fixes), ainsi qu'une diffusion à coût quasi-nul (coût marginal). De nouveaux types d'écoute émergent, les baladeurs mp3 sont ainsi basés sur l'usage intrinsèque et massif de copies d'œuvres dématérialisées. Les coûts fixes et marginaux élevés étaient la raison d'être des majors du disque, qui diminuaient leurs risques en les répartissant sur un grand nombre d'artistes. Le système actuel de rémunération abouti au paradoxe que le créateur de musique n'en touche qu'une très faible partie. Selon les chiffres du SNEP, syndicat professionnel du secteur, le parolier touche en moyenne 7%, l'interprète 19% (ce chiffre est cependant contesté par d'autres sources et la majorité des artistes seraient à 10%, les 19% concernant des artistes connus), la distribution touche 22% et la maison de disque 52%. La baisse des coûts de distribution qui peuvent maintenant être quasi-nuls et la baisse des coûts de production constituent une possibilité, soit d'une importante hausse de la rémunération de l'auteur, soit d'une baisse des prix. Il devient possible de s'affranchir de l'extorsion de la plus-value opérée par les majors du disque puisque seule reste la rémunération du travail permettant la subsistance du créateur et le remboursement des frais minimes qu'il a engagés.

L'utilisation de nouvelles technologies permet donc dans ce cas, une reprise de contrôle du créateur musicien sur son œuvre par la possession des moyens de production et de diffusion. Il peut alors se libérer des majors qui ne lui sont plus indispensables mais seulement potentiellement utiles (elles peuvent proposer aux artistes de s'occuper de tâches dont ils ne voudraient plus avoir la charge). Les majors deviennent offreuses de service pour les artistes et n'ont plus un monopole de production et de diffusion. Ceci pourrait être à l'origine d'un renouveau de la diversité culturelle, en permettant une création plus importante. Cette baisse de coût couplée à de nouvelles licences libres, copyleft ou non, permet à des artistes de diffuser leurs œuvres. Le site magnatune.com propose ainsi 3760 œuvres de 154 artistes en écoute libre depuis le site. La licence utilisée est la Creative Commons dans sa version autorisant le partage, la modification et l'utilisation non commerciale de l'œuvre. Ce qui constitue un retour à la période classique, où la création d'un autre pouvait être réutilisée dans une autre œuvre. Cette possibilité n'a disparu que quand la musique n'était plus obligatoirement jouée par des personnes physiques et qu'elle a pu devenir reproductible.

La rémunération de l'artiste s'effectue sur les utilisations commerciales de l'œuvre, pour laquelle une licence spécifique doit être achetée. Une deuxième source de revenu est la vente de la musique pour une écoute qui ne soit plus en ligne mais il demeure toujours possible de partager cette musique une fois achetée. Le consommateur choisit alors le prix qu'il souhaite acheter l'œuvre dans une

fourchette de 5 à 18$, l'artiste récupère 50% des revenus des CD, des licences et des produits dérivés, ce qui représente un montant important comparé à la part de l'ancien système. La convergence d'intérêt entre le public, qui peut écouter et partager, et l'artiste qui peut plus facilement créer et en obtenir un revenu, pourrait permettre une extension de ce système.

2) Inspiration du mode de production

- ***Encyclopédie***

L'écriture d'une encyclopédie libre est le projet à ce jour le plus avancé. Le site wikipedia dédié à cette tâche est devenu en l'espace de 3 ans la plus grande encyclopédie mondiale. Il comprend près de 270 000 articles en langues anglaises et 36 431 en français (qui n'est ouverte que depuis décembre 2001, à titre de comparaison l'encyclopaedia universalis comprend 30 000 articles). Des articles existent en 75 langues et 5 000 nouveaux apparaissent chaque jour, toutes langues confondues. Le site wikipedia est géré par la wikimedia foundation qui a mis en place d'autres sites collaboratifs qui ont toujours comme objectif la constitution de ressources informationnelles libres (dictionnaires, recueils de citations et de livres). Les articles mis en ligne sont sous licence GFDL (GNU Free Documentation Licence) qui est la licence copyleft créée initialement par Richard Stallman pour couvrir les documentations associées aux logiciels libres.

La réussite de ce projet réside dans la rencontre des mêmes conditions que celles que nous avons pues relever pour les logiciels libres. La production concerne un bien informationnel qui peut être développé et diffusé sous une forme entièrement immatérielle. Il devient alors possible d'utiliser l'organisation technique mise en place grâce à une nouvelle technologie de création collaborative de sites internet appelée Wiki, permettant à tout visiteur d'un site d'en modifier le contenu. Le projet peut alors fonctionner selon le principe des grands nombres. Plus le nombre de personnes participant est élevé, plus il est probable que l'une d'elle connaisse un sujet. Les coûts fixes se limitent à ceux du site internet et les coûts marginaux sont quasi-nuls. Une population disponible et disposant de connaissances sur le sujet sur lequel elle écrit existe.

- ***Matériel pédagogique***

La communauté enseignante dispose de grandes similitudes avec les développeurs de logiciels libres, notamment dans le rapport au savoir. Ces deux communautés ont en commun une volonté de diffusion et de partage du savoir qui, plus qu'un objectif, constitue une motivation intrinsèque.

Richard Stallman lors du lancement du projet GNU s'était inspiré des méthodes universitaires, le retour de ces méthodes vers l'éducation n'est donc qu'un juste retour des choses. En France, la production de ressources éducatives libres vient d'enseignants de mathématiques regroupés dans l'association sésamath (sesamath.net). Celle-ci sert à coordonner des actes jusqu'ici isolés et à produire la version matérielle des ressources pédagogiques libres.

Les ressources pédagogiques sont en général rassemblées dans des manuels publiés par des éditeurs privés mais le contenu est conçu par des enseignants en poste d'après leur pratique. L'éducation nationale rémunère donc des enseignants et achète aussi les livres, ce qui revient à payer deux fois pour des ressources dont elle dispose en son sein. Ce modèle date de 1811, date à laquelle l'Etat refuse d'étudier les propositions de manuels non imprimés. Les enseignants, n'ayant pas à cette époque accès aux imprimeries, ne peuvent alors plus être directement auteurs des manuels comme ils le faisaient depuis la révolution française. La rupture s'est faite sur l'impossibilité d'accès à une technologie, et elle se répare actuellement par un accès à une autre technologie. En effet, sur le modèle des logiciels libres, les ressources pédagogiques permettent une fusion du producteur et de l'utilisateur, et ainsi de se passer de l'éditeur. Les enseignants de sésamath partagent sous licence GFDL (copyleft) et leur objectif est de constituer « une réponse à la marchandisation galopante des ressources éducatives » et notamment aux négociations en cours à l'OMC (Organisation Mondiale de Commerce) sur l'AGCS (Accord General sur le Commerce des Services) qui permet la libéralisation des services. Les ressources ainsi disponibles sur sésamath sont d'ores et déjà supérieures à 4 600. Les exercices, examens et cours que l'enseignant crée pour sa classe dans le cadre de son travail sont ainsi ensuite diffusés et améliorés. Un partenariat avec le CRDP de Lille (Centre de Ressources et de Documentations Pédagogiques) permet d'éditer un CD-ROM interactif, ainsi qu'un véritable manuel vendu (30 000 exemplaires pour le moment), regroupant toutes les ressources, à un prix légèrement supérieur au prix de revient (4€ pour 115 pages, soit 22 centimes la page) que les manuels propriétaires ne peuvent égaler. Ce type de projet sur les ressources éducatives est donc appelé à se développer (1% des enseignants participant équivaut à 10 000 personnes), d'autant plus que les enseignants participant à ce type de projet peuvent dans certains cas obtenir des décharges d'enseignement pour pouvoir consacrer plus de temps à cette activité.

- ***Le matériel électronique (hardware)***

Depuis les années 1980, les composants électroniques sont dits de « synthèse », ils ne sont plus conçus et dessinés « à la main » mais par ordinateur, le schéma est maintenant généré par le synthétiseur d'après un langage de description du matériel utilisé par le concepteur. Cette évolution

a permis la poursuite rapide de l'évolution vers la miniaturisation et la complexification des composants. Elle permet aussi de diminuer radicalement le taux d'erreurs. Cette technique de production dématérialise entièrement la phase de conception et crée un « code-source » permettant la réalisation du matériel. Depuis 1990, une nouvelle innovation consiste en un rapprochement entre logiciels et matériels. Les nouvelles générations de matériel de cette époque deviennent programmables et il devient possible de modifier le fonctionnement d'un matériel par simple évolution de son logiciel intégré. Le mouvement du hardware libre naîtra en 2000, grâce à ces nouvelles possibilités techniques ainsi que de la volonté de s'affranchir des grands industriels de l'électronique. Le site Opencores.org est un des plus importants sites répertoriant tous les types de matériels libres, qui vont du processeur au contrôleur réseau, en passant par des cartes sons ou des cartes graphiques. Il est difficile d'évaluer le nombre de projets en hardware libre, cependant sur ce site, leur nombre est d'environ 250 dont une centaine de projets assez avancés pour être opérationnels. Les licences utilisées sont majoritairement la GPL mais certains projets préfèrent des licences libres non copyleft.

Certains des projets de hardware libre commencent à intéresser des industriels. Ainsi le projet Open-Risc 1000 (un microcontrôleur) a été implémenté sur silicium par Flextronics pour une application embarquée. De même le composant IP Open-Risc 1200 est utilisé par deux sociétés, Rosum et Voxi, pour leurs systèmes de positionnement embarqués. Le matériel libre est encore loin des performances des composants propriétaires complexes que sont les micro-processeurs d'Intel ou les cartes graphiques de Nvidia, mais il constitue d'ores et déjà des solutions fonctionnelles à faibles coûts et consommant peu de ressources. De plus, les sociétés souhaitant utiliser le hardware libre bénéficient de leurs complémentarités avec les logiciels libres qui leur apportent leurs supports logiciels.

Les conditions des projets de hardware libre sont très proches de celles des logiciels libres. Ils peuvent s'appuyer sur une population importante de concepteurs dont bon nombre d'étudiants et d'universitaires, pour qui ce système représente un moyen d'enseignement et d'apprentissage. La dématérialisation de la phase de conception permet l'utilisation d'un lieu de production virtuel et de bénéficier des mêmes logiciels et techniques de conception et d'organisation que ceux qui sont utilisés pour les logiciels libres. Les coûts de la conception sont réduits, puisqu'ils se limitent au site internet. Le coût le plus important résidait dans le logiciel de Conception Assisté par Ordinateur dont les versions les moins chères étaient à 50 000 €. Cependant le laboratoire LIP6, rattaché à l'Université parisienne Pierre et Marie Curie, propose maintenant un logiciel sous licence GPL permettant d'effectuer ce travail.

Il est difficile de prédire avec certitude quelle sera la réussite future du hardware libre, les similitudes avec les conditions ayant permis l'essor des logiciels libres pourraient lui permettre d'en

suivre le chemin. Le hardware libre n'en constitue pas moins une démarche intéressante et porteuse qui, couplée aux logiciels libres pourrait permettre d'envisager une informatique totalement libre à moyen terme.

Conclusion

Si les logiciels sont déjà un sujet d'étude complexe pour les théories économiques, de par certaines de leurs propriétés intrinsèques. Les logiciels libres accentuent cette difficulté théorique à saisir leur fonctionnement. Ainsi, en analysant le concept de Bazar de Eric S. Raymond, nous en déduisons une affinité de celui-ci pour les théories économiques libertariennes. Nous avons donc décidé de distinguer son analyse objective du Bazar, de ses conceptions normatives. La compréhension de ce point nous permet d'analyser la nature des licences de type open-source qu'il promeut. Grâce à cela, nous pouvons introduire un nouveau critère d'analyse, en sus de celui de la centralisation ou de la décentralisation retenue par E.S. Raymond. Il s'agit du type d'appropriation du logiciel, soit privée soit collective. Ce double critère nous permet de distinguer quatre idéaux-types d'organisation de la production de logiciels libres, résultat de l'instabilité du développement de logiciels libres. Nos deux critères, l'un social et l'autre technique, nous permettent de les qualifier de mode de production *technico-social*. Nous avons, dans la partie suivante, détaillé ces quatre modes de production et déterminé les avantages relatifs de chacun. Une fois les contours de ces quatre modes définis, nous pouvons, dans la deuxième partie, étudier comment chacun de ces quatre modes, définis jusqu'ici théoriquement, peuvent répondre à la production des deux types de logiciels, standards et spécialisés. Nous pouvons ainsi percevoir comment les acteurs se positionnent et quels sont les intérêts à effectuer un transfert entre modes selon le type de logiciel produit. Les analyses contenues dans ces trois parties nous permettent d'aboutir aux extensions opérées, depuis les logiciels libres vers d'autres biens informationnels. Nous avons pu effectuer des parallèles soit de type de licence, soit de mode de production et ainsi vérifier que les logiciels libres peuvent constituer un modèle de développement pour d'autres biens informationnels.

Nous conclurons dans le sens de Manuel Castels (1998), pour qui "pour la première fois dans l'histoire, l'esprit humain est une force de production directe, et pas simplement un élément décisif du système de production". Cette nouvelle ère de l'économie mondiale, abouti à ce que « le paradigme des technologies de l'information fournit les bases matérielles de son extension à la

structure sociale toute entière ». Il existe alors une différence fondamentale entre révolution industrielle et révolution informationnelle, alors que « l'industrialisme recherche la croissance économique, c'est-à-dire la maximisation de la production ; l'informationalisme vise au développement technologique, c'est-à-dire à l'accumulation de savoir et à la complexité croissante du traitement de l'information ». Et dans cette voie, les logiciels libres montrent que la coopération peut être plus efficace que la concurrence et ouvrent de nouveaux horizons.

Bibliographie

Thèses et Mémoires :

- Blin Nicolas, mémoire de sciences de l'information et de la communication, « L'usage de Flash et la standardisation des formats de publication web », 2002.

- Boyer Antoine, mémoire de Science Politique, « Étude du cyber-Mouvement du Logiciel Libre », 2003.

- Horn François, thèse d'économie sur « L'économie du logiciel », 2000.

- Jullien Nicolas, thèse d'économie sur « l'impact du logiciel libre sur l'industrie informatique », 2001.

- Labbe Frédéric, mémoire d'ingénieur en informatique, « Suite bureautique, les enjeux d'une alternative », 2003.

- Phan Denis, thèse d'économie sur les « Régimes et changements structurels. essais d'économie historique », 2002.

Ouvrages :

- Benasayag Miguel et Aubenas Florence, « Resister, c'est créer », éditions La Découverte sur le vif, 2003, 120 pages.

- Blondeau Olivier et Latrive Florent (ouvrage collectif dirigé par), « Libres enfants du savoir numérique », Editions de l'éclat, 2000, 504 pages.

- Castells Manuel, « La société en réseau », Fayard, 1998

- Jullien Nicolas, Clément-Fontaine Mélanie, Dalle Jean-Michel, Rapport final « Projet RNTL :

nouveaux modèles économiques, nouvelle économie du logiciel », 2003, 206 pages.

- Lévèque François et Menière Yann, « Economie de la propriété », La Découverte, 2003

- Rivière Philippe, chapitre 16 de « La santé mondiale, entre racket et bien public », coordonné par François-Xavier Verschave, à paraître.

- Smets Jean-Paul et Faucon Benoît, « Logiciels Libres – Liberté, égalité, business », Éditions Edispher, 1999.

- Viveret Patrick, rapport « Reconsidérer la richesse », 2002.

- Williams Sam, « Free as in Freedom : Richard Stallman's Crusade for Free Software », édition O'reilly, 2002, 240 pages.

Articles d'universitaires :

- Archambault Jean-Pierre, « Economie du savoir : Coopération ou concurrence ? », Médialog n°48, décembre 2003.

- Archambault Jean-Pierre, « l'édition scolaire au temps du numérique », Médialog n°41 septembre 2001.

- Archambault Jean-Pierre, « Vers de nouveaux rapports entre auteurs et éditeurs ? : Les auteurs numériques investissent la Toile », Médialog n°46, mai 2003.

- Bernard Lang, « Ressources Libres et Indépendance Technologique dans les Secteurs de l'Information », Actes du Colloque Inforoutes et Technologies de l'Information, 1997.

- Bezroukov Nikolaï, « A second look at the Cathedral and the Bazaar », First Monday, 1999.

- Bezroukov Nikolaï, « BSD vs. GPL: A Framework for the Social Analysis », 2002.

- Bezroukov Nikolaï, « open source software as a special type of Academic Research (Critique of vulgar raymondism), First Monday, 1999.

- Blondeau Olivier, « Genèse et subversion du capitalisme à l'ère informationnelle, Linux et les logiciels libres », dans La Pensée, n° 317, Janvier-Mars 1999.

- Blondeau Olivier, « Le style bazar. Vers un nouveau paradigme de construction des savoirs », Communication au colloque « Multimédia et construction des savoirs », mai 1999.

- Blondeau Olivier, « Tous producteurs, tous consommateurs ? », Contribution au séminaire européen. "Les politiques de protection et de valorisation des patrimoines dans la mondialisation",

février 2003.

- Boutang Yann Moulier, « Nouvelles frontières de l'économie politique du capitalisme cognitif », La rencontre NTIC de Mantes la Jolie des 26 et 27 mars 2002 : Vers un paradigme des sociétés de création et de coopération ?

- Brooks Frederick P., « Le mythe du mois-homme : Essais sur le génie logiciel », International Thomson, Publishing, 1996, 276 p., in Horn François (2000), thèse « L'économie des logiciels » et in Raymond Eric S., « La cathédrale et le Bazar », http://www.catb.org/~esr/, 1998.

- Browne Christopher B., « Linux et le développement décentralisé », First Monday, 1998

- Cavalier Forrest J., « Some Implications of Bazaar Size », http://www.mibsoftware.com/bazdev/, 1998.

- Challet Damien et Le Du Yann, « Closed Source versus Open Source in a Microscopic Model of Software Bug Dynamics », 2003.

- Cohendet Patrick, Creplet Frederic, Dupouët Olivier, « Communities of Practice and Epistemic Communities: A Renewed Approach of Organisational Learning within the Firm ».

- Cox Alan, « La cathédrale, le bazar et le conseil municipal », http://www.catb.org/~esr/, 1998.

- Dalle Jean-Michel, « Manuels “ Libres ” ou “ Napster ” Educatif ? », Séminaire “ Propriété intellectuelle et économie des biens informationnels ”, 23 Mars 2001.

- Dang-Nguyen G. et Pénard T., « Economie de l’Internet et coopération en réseau », 2003. http://perso.univ-rennes1.fr/thierry.penard/biblio/Ecointernet0.pdf

- Delsalle Sébastien, « Vers une cité du libre : le modèle théorique de la cité par projets », freescape.org, mars 2004.

- Genthon Christian & Phan Denis, « Les Logiciels Libres : un nouveau modèle ? », 2000, http://www-eco.enst-bretagne.fr/Recherche/Biblio/Denis/gentphan.pdf

- Ghosh Rishab Aiyer, « Les marchés « marmite » : un modèle économique pour le commerce de biens et de services gratuits sur l'Internet », First Monday, 1998.

- Giarini Orio, Stahel Walter R., « Les limites du certain : affronter les risques dans une nouvelle économie de service», 1990 in Horn François, thèse « L'économie des logiciels » 2000.

- Giraud Pierre-Noël, « Un spectre hante le capitalisme : la gratuité », Le Monde, 5 mai 2004.

- Husson Michel, « Sommes-nous entrés dans le capitalisme cognitif ? », Critique communiste n° 169-170, été-automne 2003.

- Jullien Nicolas, « Linux, la convergence du monde Unix et du monde PC ? », Revue Terminal,

1999.

- Jullien Nicolas, Horn François, Demazière D., « Le travail des développeurs de logiciels libres », Autour du libre, 30 avril 2004.

- Ko Kuwabara, « Linux: A Bazaar at the Edge of Chaos », First Monday, 2000

- Lakhani Karim R. et Wolf Robert G., « Why Hackers Do What They Do: Understanding Motivation Effort in Free/Open Source Software Projects », MIT Sloan School of Management, 2003.

- Lancashire David, « Code, culture and cash : The fading altruism of open source development », First Monday, 2001.

- Lerner J., Tirole J., « The simple economics of Open Source », mimeo, Harvard Business School, 2000.

- Meltz Raphaël, « Logiciels libres », R de réel volume L, 2002.

- Mounier Pierre, « Le libre pour les publications scientifiques : pertinence et limites d'un modèle commun », colloque Autour du libre, 21 mai 2003.

- OCDE, « Le génie logiciel : un défi pour l'action gouvernementale », OCDE (PIIC Politiques d'Information, d'Informatique et de Communications n°26), 1991, 66 pages, in Horn François, thèse « L'économie des logiciels » 2000.

- Perens Bruce, « La définition de l'Open Source », http://www.linux-france.org/article/these/osd/fr-osd.html, 1998.

- Philip Ball, « Openness makes software better sooner », Nature, 25 Juin 2003.

- Raymond Eric S., « À la conquête de la noosphère », http://www.catb.org/~esr/, 1998.

- Raymond Eric S., « Goodbye, "free software"; hello, "open source" », http://www.catb.org/~esr/, 1998

- Raymond Eric S., « La revanche des hackers », http://www.catb.org/~esr/, 1998.

- Raymond Eric S., « Une brève histoire des hackers », http://www.catb.org/~esr/, 1998.

- Rideau Faré, « About Eric S. Raymond's Articles », http://fare.tunes.org/articles/about_esr.html, 2003.

- Rishab A. Ghosh, Ruediger Glott, Bernhard Krieger, Gregorio Robles, « FLOSS Survey and Study », 2002.

- Rivière Philippe, « La pilule et l'ordinateur: quelles passerelles entre le mouvement des logiciels

libres et les campagnes pour l'accès aux médicaments ? », colloque Autour du libre, 21 mai 2003.

- Shirky Clay, « The Interest Horizons and the Limits of Software Love », shirky.com, 1999.

- Stallman Richard, « Le système d'exploitation du projet GNU et le mouvement du logiciel libre », http://www.linux-france.org/article/these/gnuproject/fr-thegnuproject.html, 1998.

- Stallman Richard, interview donnée à Multitudes n°1, février 2000.

- Torvald Linus , « What motivates free software developers ? », First Monday, 1998.

- Torvald Linus, « The Pragmatist of Free Software », interview KDE.org, 1997.

- Vercellone Carlo, « Division internationale du travail, propriété intellectuelle et politique de développement à l'heure du capitalisme cognitif », 3èmes journées d'étude « Approches économiques et pluridisciplinaires du Patrimoine » Université de Reims, 2003

- Vercellone Carlo, Dieuaide Patrick et Paulré Bernard, « le capitalisme cognitif », 3èmes journées d'étude « Approches économiques et pluridisciplinaires du Patrimoine » Université de Reims, 2003.

- Vercellone Carlo, Dieuaide Patrick, Lojkine Jean et Husson Michel, « Table-ronde sur le capitalisme cognitif », Regards n°89, avril 2003.

- Viseur Robert, « Apache : analyse d'un succès passé et présent », ecocentric.be, 2003

- Viseur Robert, « Aspects économiques et business models du logiciel libre », Texte de la journée du Logiciel Libre, 11 septembre 2003.

- Viseur Robert, « La dynamique open source », ecocentric.be, 2003

- Weber Steven, « The Success of Open Source », interview dans Ubiquity volume 5 issue 11, 12-18 mai 2004.

- Wiener Lauren Ruth, « Les avatars du logiciel : leçons à tirer de quelques jolis fiascos informatiques », Addison-Wesley (Mutations technologiques), 1994, in Horn François, thèse « L'économie des logiciels » 2000.

- Zimmermann Jean-Benoît, « Logiciel et propriété intellectuelle : du Copyright au Copyleft », freescape.eu.org, 1999.

- Zimmermann Jean-Benoît, « Logiciel Libre et marchandisation : un problème d'incitation pour les développeurs », colloque Autour du libre, mai 2003.

- Zimmermann Jean−Benoît et Foray Dominique, « L'économie du Logiciel Libre: organisation coopérative et incitation à l'innovation », freescape.eu.org, 2001.

Articles :

- Becker David, « Se habla open source ? », CNET News.com 16 février 2004.

- Bénassy Odile, « La méthode de commandes d'Amazon brevetée en Europe », http://swpat.ffii.org/neuf/03/amaz0818/index.fr.html, 18 août 2003.

- Bruce Perens, « UserLinux: Get ready for a bulletproof distribution », interview dans Linux world, 16 mars 2004.

- Bruce Perens, « UserLinux: Repairing the Economic Paradigm of Enterprise Linux », Userlinux.com, 2004.

- Campbell-Kelly Martin, « Une histoire de l'industrie du logiciel », Vuibert Informatique, 2003

- Cassia Fernando, « When Open Source goes hardware », The Inquirer, 11 décembre 2003.

- De Icaza Miguel, « Defining the Game » , http://primates.ximian.com/~miguel/activity-log.php, 24 avril 2004.

- Delbecq Denis, « La revue en ligne qui fait trembler les payantes », Liberation, 06 mai 2004.

- Festa Paul, « Developers gripe about IE standards inaction », CNET News.com, 9 Octobre 2003.

- Foster Glynn « GNOME Foundation / Mozilla Foundation Meeting Minutes », foundation-list gnome.org, lundi 26 avril 2004.

- Foucart Stéphane, « Le "libre accès" aux résultats de la recherche bouleverse le monde des revues savantes », Le Monde, 16 avril 2004.

- Gates Bill, « Lettre ouverte aux "Hobbyistes"», freescape.eu.org, 1976.

- Guillemin Christophe, « E-gouvernement: la Commission européenne milite pour les standards ouverts », ZDNet France, 22 juillet 2003.

- Himmelsbach Vawn, « Asian Governments Start to Speak the Same Language on Linux Implementations », linuxinsider.com, 2004.

- Ian Murdock, « Reconsidering Linux », cnet.com, 30 juillet 2003.

- Koenig John, « Seven open source business strategies for competitive advantage », IT's manager journal, 14 mai 2004.

- Kohn Alfie, « Studies Find Reward Often No Motivator : Creativity and intrinsic interest diminish if task is done for gain », Boston Globe, Janvier 1987.

- Lagace Martha, « The Simple Economics of Open Source », 2000.

- Lawton Graham, « Open Cola : premier soda en licence libre », The new west indian, 18 juillet

2002.

- McKie Robin and Thorpe Vanessa, « Digital Domesday Book lasts 15 years not 1000 », The Observer, 3 mars 2002.

- Nitot Tristan et Boudreau Denis, « Pourquoi les standards du W3C ? », Openweb.eu.org, 21 mars 2003.

- Rivlin Gary, « Leader of the Free World :How Linus Torvalds became benevolent dictator of Planet Linux, the biggest collaborative project in history », Wired n°11.11, novembre 2003.

- Roucairol Gérard, « Le RNTL a donné une assise industrielle à plus de cent projets de recherche », 01 informatique, 2004.

- Vaughan-Nichols Steven J., « Longhorn's Real Job: Trying to Gore Linux », eweek.com, 29 avril 2004.

Conférences :

- Aurray Nicolas, ENST, cycle éducation du salon Solution Linux, Paris, 3-4-5 février 2004.

- Marchand Eric et Archambault Jean-Pierre, conférence « Brevets et Appropriation du Bien Commun. Logiciels et enjeux du marché mondial », École Supérieure en Sciences Informatiques – Université de Nice Sophia Antipolis, 15 avril 2004.

- Stallman Richard, GNU FSF, keynote du salon Solution Linux, Paris, 3-4-5 février 2004.

Personnes ressources :

- Archambault Jean-Pierre, CNDP, FSU, ATTAC.

- Gautier Sophie, conseil de la communauté OpenOffice.org.

- Hache Sébastien, sésamath.

- Lacage Mathieu, fondation GNOME.

- Males Simon, fondation Mozilla.

- Nitot Tristan, fondation Mozilla Europe.

- Riviere Philippe, Le Monde Diplomatique.

- Stallman Richard, GNU, Free Software Foundation.

- Veillard Daniel, fondation GNOME.
-

Sites internet :

- http://opencollector.org
- http://www.mozilla.org/foundation
- http://foundation.gnome.org/
- http://www.osdl.org/
- http://council.openoffice.org/
- http://www.plos.org/
- http://www.magnatune.com/
- http://fr.wikipedia.org/
- http://sesamath.net/
- http://artlibre.org/

Annexes

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you

can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another

language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

* a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

* b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

* c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or

else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

* a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

* b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and

2 above on a medium customarily used for software interchange; or,

* c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME

THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Table des matières