

Sommaire

Introduction.....	3
Partie I : Présentation du logiciel libre	5
I. Introduction	6
II. Définition du logiciel libre	6
1. Définition du logiciel.....	6
2. Logiciel libre	7
3. Les différents types de licences de logiciels.....	9
a. Fondement économique.....	9
b. Les différentes catégories de logiciels.....	10
4. Polémique Open Source - Freesoftware	13
b. Introduction	13
c. La philosophie « Open Source ».....	14
d. La philosophie «Freesoftware ».....	14
e. Conclusion.....	15
III. Cadre juridique.....	15
1. Introduction	15
2. La Free Software Foundation (FSF).....	16
3. La licence GPL (Genral Public License)	17
4. Le projet GNU.....	18
IV. Bref historique.....	19
1. Le modèle Unix	19
2. Les premiers systèmes libres	20
3. L'arrivée de Linux	21
V. Conclusion.....	23
Partie II : Economie du logiciel libre.....	25
I. Introduction	26
II. Economie du logiciel.....	26
1. Aspects généraux.....	26
a. Introduction	26
b. Economie du logiciel propriétaire commercial	27
2. Les bonnes propriétés	29
III. Modèle de production du Libre	30
1. Introduction	30
2. La cathédrale et le Bazar, Eric S. Raymond 1998	30
a. Introduction	30
b. La cathédrale et le bazar	31
c. Fonctionnement du bazar.....	31
d. Les facteurs de la réussite d'un projet Open Source.....	32
3. Pourquoi le modèle Open Source est sensé, Linus Torwalds, 2001	33
a. Introduction	33
b. Rappel des caractéristiques du bazar	33
c. Le modèle de la recherche scientifique et les effets secondaires insoupçonnées de la « philosophie Open Source »	34
d. Quand une entreprise est l'initiatrice d'un projet Open Source	35
e. Les conséquences pour une entreprise de s'ouvrir.....	35
f. Intérêts et limites du texte.....	36

4.	Conclusion.....	36
IV.	Les avantages du logiciel libre	37
1.	Introduction	38
2.	Les avantages	39
a.	Introduction	39
b.	Fonctionnalité	39
c.	Rentabilité.....	40
d.	Efficacité.....	40
e.	Fiabilité.....	40
f.	Compatibilité avec les standards.....	40
g.	Garantie de fonctionnement.....	41
h.	Garantie de liberté	41
i.	Pérennité	42
3.	La problématique des coûts	42
V.	Conclusion.....	44
Partie III : Typologie des Business Models du logiciel libre.....		45
I.	Introduction	46
II.	Présentation général.....	47
1.	La notion de Business Model	47
2.	Les points communs des Business Models du Libre	48
III.	Typologie des Business Models du Libre	49
1.	Le modèle non business.....	49
a.	Présentation du modèle	49
b.	Apache.....	51
2.	Le modèle du service et du hardware	54
a.	Présentation du modèle	54
b.	Alcove.....	56
3.	Le modèle des distributions Linux	58
a.	Définition d'une distribution Linux	58
b.	Présentation du Business Model	60
c.	Mandrake	62
4.	Le modèle défensif	65
a.	Présentation du modèle	65
b.	Netscape	66
5.	Le modèle des logiciels « à façon ».....	68
a.	Présentation du modèle	68
b.	IDX-PKI	71
6.	Conclusion.....	72
IV.	Modèle de croissance global	73
1.	Système d'interaction entre les Business Models.....	73
a.	<i>Le modèle non Business</i> : la clef de voûte du Libre	73
b.	Interaction entre les projets.....	75
c.	Conclusion	76
2.	Viabilité du modèle global de croissance du Libre.....	76
a.	Le marché des entreprises.....	77
b.	Le marché grand public	77
c.	Conclusion.....	80
Conclusion.....		81
Bibliographies		84

Introduction

Introduction Générale

Introduction

Depuis quelques mois, la littérature économique voit fleurir de nombreux articles sur le logiciel libre. Mais ce phénomène du logiciel libre ne fascine pas uniquement les économistes. Des acteurs majeurs de l'industrie informatique font désormais le pari du logiciel libre, c'est le cas notamment d'IBM ou de Sun. La presse informatique ne cesse d'évoquer le phénomène Linux, fer de lance des logiciels libres. Même le directeur général de Microsoft, Steve Ballmer, annonce que le seul concurrent sérieux de son entreprise est Linux et les logiciels libres. Le logiciel libre serait-il devenu le nouvel Eldorado de l'industrie informatique, après la déconvenue de la nouvelle économie et de l'échec des start-up Internet ?

Pour étudier ce phénomène du logiciel libre nous allons nous pencher plus précisément sur les Business Models des entreprises qui vivent du logiciel libre. Ainsi la problématique de ce mémoire est de *comprendre les différents types de Business Models du logiciel libre, pour essayer d'appréhender la viabilité de cette nouvelle approche de l'industrie du logiciel*. Pour répondre à cette problématique nous allons procéder en trois étapes :

Dans la première partie de ce mémoire, nous allons présenter la notion de logiciel libre, pour cela nous verrons ses aspects techniques, juridiques, et historiques.

Dans un second temps, nous nous intéresserons à l'économie du logiciel libre. Après une rapide introduction sur l'économie du logiciel, nous verrons que le logiciel libre possède des particularités qui permettent un mode de production originale. Nous verrons ensuite que ce mode de production est à l'origine des qualités du logiciel libre.

Enfin, dans un troisième mouvement, nous répondrons à la problématique de ce mémoire, en présentant une typologie des Business Models du logiciel libre.

Partie I

Présentation du logiciel libre

I. Introduction

Le logiciel libre est source de nombreuses confusions. En particulier en anglais, logiciel libre se dit freesoftware ce qui engendre la confusion du fait de la double signification en anglais du mot free, à la fois liberté et gratuité. En même temps, le monde du logiciel libre est souvent perçu comme un monde de marginaux et de doux rêveurs.

Avant d'étudier en détail les business models du logiciel libre, il est important de rappeler quelques éléments de base pour la compréhension du phénomène du Libre¹. On va ainsi s'attacher à définir exactement la notion de logiciel libre. On va aussi se pencher sur les aspects techniques et juridiques du logiciel libre. Enfin, nous allons rappeler brièvement l'histoire du logiciel libre.

II. Définition du logiciel libre

1. Définition du logiciel

De façon formelle, un logiciel se définit comme « *l'ensemble des programmes, des procédés, des règles et éventuellement de la documentation, relatifs au fonctionnement d'un ensemble de traitement de données* »². Plus généralement, on pourra réduire ici cette définition à l'ensemble des instructions logiques indispensables au fonctionnement d'un ordinateur. Avec la démocratisation de l'informatique et la démultiplication de ses effets au travers de l'Internet, l'importance des logiciels n'a fait que croître, depuis leur apparition dans les années 60 jusqu'à la société de l'information que nous connaissons aujourd'hui.

¹ Déf : Système de production et de diffusion des logiciels libres (Jullien, 2001)

² <http://www.ccip.fr/irpi/faq/logiciel/definition.htm>

D'un point de vue technique, un logiciel est constitué d'un *code source* et d'un *code exécutable*. Un petit détour par une analogie musicale permet d'éclairer ces termes. On peut considérer le code source d'un logiciel comme la partition de celui-ci, et le code exécutable comme sa version enregistrée. Une partition peut être jouée sur un piano, une flûte ou par un orchestre philharmonique. Mais si la même partition est pressée sur un disque après un concert de harpe, l'acheteur ne peut rien modifier et ne pourra pas jouer le morceau à la guimbarde ou moduler l'interprétation.

Le passage de l'une à l'autre version s'opère grâce à une « compilation », qui traduit le code source (des lignes écrites en usant de langages de programmation maîtrisés par les informaticiens) en code exécutable (uniquement compréhensible par l'ordinateur). Avant d'être vendue dans le commerce, une version en code source d'un logiciel est construite patiemment par les programmeurs d'une entreprise ou d'une communauté.

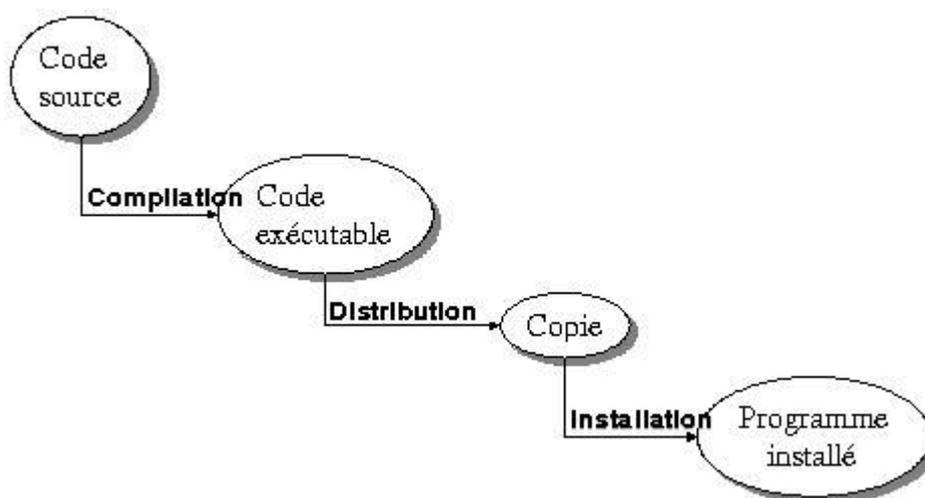


Fig 1. Architecture d'un logiciel
Source Rapport Carcenac³

2. Logiciel libre

La définition et la description de l'architecture d'un logiciel que nous venons de présenter, nous permettent de comprendre ce qu'est un logiciel libre. En effet, un logiciel

³ <http://www.internet.gouv.fr/francais/textesref/rapcarcenac/sommaire.htm>

libre est un logiciel qui est livré avec son « code source » alors qu'un logiciel « propriétaire » est un logiciel livré sous forme « d'exécutable ».

La Free Software Foundation (FSF)⁴ propose une définition complète qui sert de référence dans le monde du Libre, basée sur 4 libertés fondamentales.

L'expression « Logiciel libre » fait référence à la liberté pour les utilisateurs d'exécuter, de copier, de distribuer, d'étudier, de modifier et d'améliorer le logiciel. Plus précisément, elle fait référence à quatre types de liberté pour l'utilisateur du logiciel:

- **La liberté d'exécuter le programme, pour tous les usages (liberté 0).**
- **La liberté d'étudier le fonctionnement du programme, et de l'adapter à vos besoins (liberté 1). Pour ceci l'accès au code source est une condition requise.**
- **La liberté de redistribuer des copies, donc d'aider votre voisin (liberté 2).**
- **La liberté d'améliorer le programme et de publier vos améliorations, pour en faire profiter toute la communauté (liberté 3). Pour ceci l'accès au code source est une condition requise.**

Un programme est un logiciel libre si les utilisateurs ont toutes ces libertés. Ainsi, on est libre de redistribuer des copies, avec ou sans modification, gratuitement ou non, à tout le monde, partout. Être libre de faire ceci signifie (entre autre) que l'on n'a pas à demander ou à payer pour en avoir l'autorisation de l'utiliser.

On doit aussi avoir la liberté de faire des modifications et de les utiliser à titre personnel dans son travail ou ses loisirs, sans en mentionner l'existence. Si on publie les modifications, on n'est pas obligé de prévenir quelqu'un en particulier ou de le faire d'une manière particulière.

La liberté d'utiliser un programme est la liberté pour tout type de personne ou d'organisation de l'utiliser pour tout type de système informatique, pour tout type de tâche et sans être obligé de communiquer ultérieurement avec le développeur et tout autre entité spécifique.

La liberté de redistribuer des copies doit inclure les formes binaires ou exécutables du programme (tout comme le code source) à la fois pour les versions

⁴ <http://www.fsf.org>

modifiées ou non modifiées du programme. Il y a une exception s'il n'y a pas moyen de produire une version binaire ou exécutable, mais le public doit avoir la liberté de distribuer de telles formes s'ils ont un moyen d'en produire.

Pour avoir la liberté d'effectuer des modifications et de publier des versions améliorées, on doit avoir accès au code source du programme. Par conséquent, l'accessibilité du code source est une condition requise pour un logiciel libre.

Pour que ces libertés soient réelles, elles doivent être irrévocables tant que l'on a rien fait de mal ; si le développeur du logiciel a le droit de révoquer la licence sans que l'on ait fait quoi que ce soit pour le justifier, le logiciel n'est pas libre.

Cependant, certains types de règles sur la manière de distribuer le logiciel libre sont acceptables tant que ces règles ne rentrent pas en conflit avec les libertés fondamentales. Par exemple, le copyleft (pour résumer très simplement) est une règle qui établit que lorsque les programmes sont distribués, personne ne peut ajouter de restrictions pour retirer les libertés fondamentales au public. Cette règle ne rentre pas en conflit avec les libertés fondamentales ; en fait, elle les protège.

3. Les différents types de licences de logiciels

a. Fondement économique

L'économie logicielle présente des caractéristiques particulières par rapport à d'autres secteurs plus traditionnels, comme nous le verrons plus en détail dans la deuxième partie. En effet, la valeur d'un logiciel reste essentiellement intellectuelle, vu que sa production et encore plus sa distribution ne demandent que des investissements réduits par rapport aux coûts liés au développement, qui emploie parfois des milliers de programmeurs simultanément. Le fort contenu intellectuel des logiciels est généralement protégé par de nombreuses licences, accordant différents droits selon le but visé par la version concernée. Un shareware qui aura pour but de faire découvrir le logiciel verra sa distribution et sa copie autorisées, mais pas son utilisation au delà d'une période d'essai. A l'opposé un logiciel propriétaire commercial, comme un système d'exploitation, un progiciel spécialisé ou un format de stockage de données, pourront se voir très contrôlés au niveau de la distribution et même de l'utilisation en entreprise. Par exemple utiliser

certaines logiciels propriétaires sur plus de machines que ne l'autorise sa licence, ou avec des périphériques particuliers peut entraîner l'annulation du contrat, en clair l'interdiction d'utiliser plus longtemps le logiciel.

Ces contrats de licence ont d'ailleurs de nombreux aspects très particuliers. Les éditeurs logiciels cèdent des licences d'exploitation propriétaires permettant à leurs clients d'utiliser sous certaines conditions leur produit. En aucun cas les clients ne possèdent le logiciel, seulement le droit de l'utiliser. De plus, aucune garantie quant à son fonctionnement n'est présente, et en cas de perte de données ou de panne liée au logiciel, le client ne pourra se retourner contre l'éditeur, comme le montre un extrait de licence tout à fait courant. Ce qui pourrait sembler abusif dans d'autres secteurs d'activité est ici justifié juridiquement par le caractère « artistique » des logiciels, qui en tant qu'œuvres ne peuvent donc être associées à aucune garantie. La responsabilité de l'éditeur n'est ainsi jamais engagée, seule sa caution morale certifiant que le logiciel est fiable peut rassurer le client.

b. Les différentes catégories de logiciels

- **Logiciel Libre**

« Logiciel libre » ne signifie pas « non commercial ». Un logiciel libre peut être disponible pour un usage commercial. Le développement commercial de logiciels libres n'est plus l'exception ; de tels programmes sont des logiciels commerciaux libres.

- **Copyleft**

Dans le projet GNU de Richard Stallman, le concept de « **copyleft** » a été introduit pour protéger les quatre libertés fondamentales. Pour les auteurs de ce projet, la meilleure manière de rendre un logiciel libre est de le « copylefter », c'est à dire de le distribuer dans le domaine public, **sans copyright**. Cela autorise les gens à partager le programme et leurs améliorations. Le **copyleft** indique que quiconque les redistribue, avec ou sans modifications, **doit** aussi transmettre la liberté de les copier et de les modifier. Le copyleft garantit cette liberté partout et à tous les utilisateurs.

- **Logiciel Open Source**

Le terme « open source » exprime quelque chose de proche (mais pas tout à fait identique) au « logiciel libre » en ce sens que le code source est mis librement en circulation tout en permettant à son auteur de conserver son copyright..

- **Logiciel du domaine public**

Logiciel du domaine public veut dire logiciel non soumis aux droits d'auteur. C'est un cas particulier de logiciel libre "non-copylefté", ce qui veut dire que certaines copies, ou certaines versions modifiées, peuvent ne pas être du tout libres. Le terme, «domaine public» est un terme légal qui signifie précisément que le logiciel n'est pas « soumis au copyright ».

- **Logiciel copyleft (sous gauche d'auteur)**

Le logiciel sous copyleft (littéralement, gauche d'auteur) est un logiciel libre, dont les conditions de distribution interdisent aux nouveaux distributeurs d'ajouter des restrictions supplémentaires lorsqu'ils redistribuent ou modifient le logiciel. Ceci veut dire que chaque copie du logiciel, même si elle a été modifiée, doit être un logiciel libre.

Par opposition au copyright, ce type de licence (la plus courante du Libre) vise à éviter l'apparition de restrictions quant au statut du logiciel au cours de son développement. Une fois qu'un logiciel est copylefté, ses évolutions ne pourront être revendiquées par qui que ce soit, et son code source restera toujours ouvert et librement modifiable. Le concept de copyleft est un concept général. Pour l'appliquer à un programme, il existe un ensemble de termes relatifs à la distribution, ce qui fait que concrètement, il existe plusieurs façons d'écrire les conditions de distribution. Seul l'objectif reste le même.

- **Logiciel libre non-copylefté**

Le logiciel libre non-copylefté est diffusé par son auteur qui donne la permission de le redistribuer et de le modifier, mais aussi d'y ajouter d'autres restrictions. Si un programme est libre, mais non-copylefté, alors certaines copies ou versions modifiées peuvent ne plus être libres du tout. Une société informatique peut compiler ce programme, avec ou sans modifications, et distribuer le fichier exécutable sous forme de produit logiciel propriétaire. Le Système X-Window illustre bien ce cas.

- **Logiciel couvert par la GPL**

La GNU GPL (Licence Publique Générale GNU) est un ensemble spécifique de conditions de distribution pour copylefter un programme. Le projet GNU l'utilise comme conditions de distribution de la plupart des logiciels GNU.

- **Logiciel semi-libre**

Le logiciel semi-libre est un logiciel qui n'est pas libre, mais qui s'accompagne de la permission pour les personnes physiques de l'utiliser, de le copier, de le distribuer, et de le modifier (y compris pour la distribution des versions modifiées) dans un but non lucratif. Un logiciel semi-libre est toujours mieux qu'un logiciel propriétaire, mais cela pose toujours des problèmes.

- **Logiciel propriétaire**

Le logiciel propriétaire est un logiciel qui n'est ni libre, ni semi-libre. Son utilisation, sa redistribution ou sa modification sont interdites, ou exigent une autorisation spécifique, ou sont tellement restreintes que cela ne peut être fait librement.

- **Freeware** (graticiel)

Le terme « freeware » n'a pas de définition claire communément acceptée, mais elle est utilisée couramment pour des paquetages qui autorisent la redistribution mais pas la modification (et dont le code source n'est pas disponible). Ces paquetages ne sont pas des logiciels libres.

- **Shareware** (partagiciel)

Le partagiciel est un logiciel qui s'accompagne de la permission de redistribuer des copies, mais qui mentionne que toute personne qui continue à en utiliser une copie est obligée de payer des royalties.

- **Logiciel commercial**

Le logiciel commercial est un logiciel développé par une entreprise dont le but est de gagner de l'argent sur l'utilisation du logiciel. « Commercial » et « propriétaire » ne sont pas synonymes ! La plupart des logiciels commerciaux sont propriétaires, mais il y a

des logiciels libres commerciaux, et il y a des logiciels non-commerciaux non-libres (freewares).

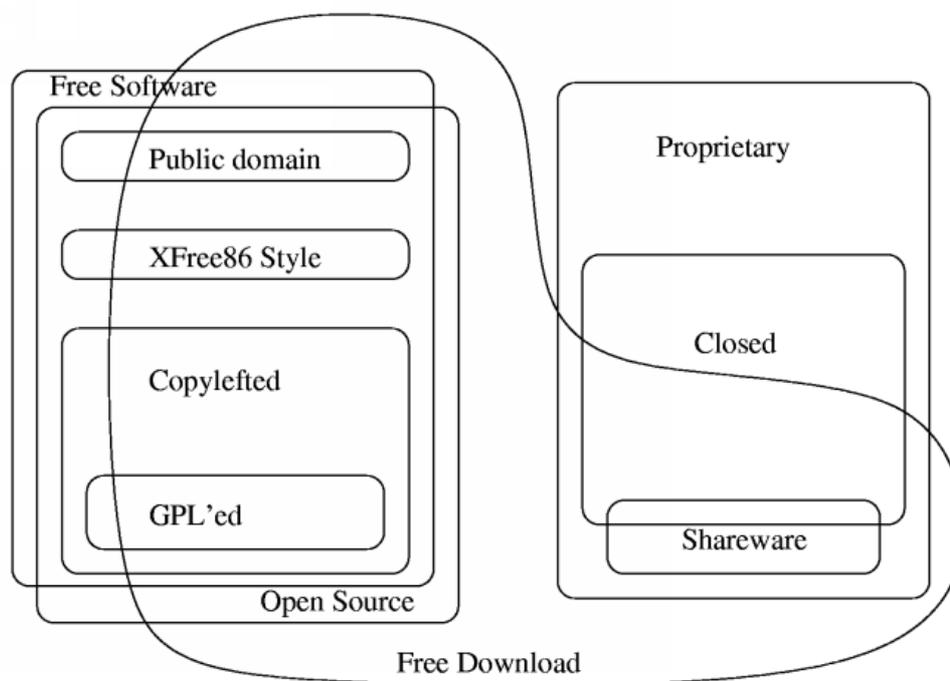


Fig 2. Les différentes catégories de logiciels.
Diagramme de Hung Chao-Kuei, *source* FSF⁵

4. Polémique Open Source - Freesoftware

b. Introduction

On trouve souvent dans la littérature la notion de logiciel Open Source pour parler des logiciels libres. La notion de logiciel Open Source est apparue du fait de la confusion que le terme anglais de Freesoftware faisait naître. En effet, le mot free se traduit indifféremment par libre ou par gratuit. Certains, dans le monde du Libre, en particulier Linus Torwalds (père du célèbre logiciel Linux), ont choisi de parler de logiciel Open Source pour éviter la confusion sur la gratuité qui est selon lui préjudiciable pour le Libre

⁵ <http://www.fsf.org/philosophy/categories.fr.html>

car elle éloignerait les investisseurs de ce mouvement. Au contraire, certains irréductibles rassemblés autour de Richard M. Stallman, autre grande figure du libre et père du projet GNU et de la FSF, préfèrent rester fidèles à la notion de Freesoftware. Il s'agit de deux courants de pensées antagonistes mais qui font tous deux partie de la famille du Libre (le terme français est très souvent employé en anglais pour regrouper les partisans du concept de Freesoftware et d'Open Source). Ces deux courants ont deux philosophies légèrement différentes du Libre.

c. La philosophie « Open Source »

Pour les mouvements Open Source, la priorité est la fabrication de logiciels les plus parfaits possibles. La technique est simple : lors de l'utilisation d'un logiciel libre, si un bug est rencontré, il peut être notifié par e-mail au développeur (ou être corrigé si on en est capable), et très vite un correctif sera disponible sur l'Internet. De la même façon, chacun peut apporter sa contribution en améliorant le programme.

Avec un produit comme Word, l'absence du code source rend tout cela simplement impossible. De plus, la hotline de Microsoft se moquera totalement de savoir qu'un bug a été trouvé ou que l'on souhaiterait ajouter une fonction à un logiciel. Pour le mouvement Open Source, la priorité est donc d'avoir des logiciels de qualité irréprochable. Ce n'est pas un défi pour l'argent mais un défi intellectuel. Nous reviendrons plus en détails sur les incitations à développer un logiciel libre dans la deuxième partie de ce mémoire. Le mouvement Open Source s'intéresse d'abord à l'aspect technique des choses et pas à la liberté. Ce terme Open Source n'est pas très clair car un logiciel Open Source, s'il a ses sources en accès libre, n'est pas nécessairement libre pour autant. S'il est accompagné d'une licence trop restrictive, l'accès aux sources le rend tout aussi propriétaire qu'un logiciel propriétaire de base.

d. La philosophie «Freesoftware »

Le mouvement Freesoftware n'a, bien sûr, rien contre les logiciels de qualité mais a un but plus philosophique.

Selon Richard M. Stallman « *les logiciels libres sont une question de liberté, pas de prix* »

Le but est de faire des logiciels qui rendent l'utilisateur libre : libre de copier, distribuer, étudier et modifier. Un développeur de logiciels libres peut être payé pour son travail. Le fruit de son travail, même s'il lui apporte un intérêt particulier (financier), apportera inévitablement une amélioration au bien commun. De plus, concevoir un logiciel libre qui est la réplique d'un logiciel propriétaire (par exemple faire un remplacement à MS Word) permet la désaliénation des utilisateurs et les fait ainsi tendre vers plus de liberté. S'il s'avère que les logiciels libres sont de meilleure qualité, ce n'est pas le principal but recherché, il faut garder à l'esprit que la liberté reste la finalité. Par ailleurs, aussi bien dans le mouvement Freesoftware qu'Open Source, on a affaire à des programmeurs dont la passion décuple les forces.

e. Conclusion

Cette digression sur la polémique Open Source – Freesoftware, a plusieurs intérêts dans le cadre d'une réflexion sur les « Business Models du logiciel libre ». Elle permet de comprendre l'utilisation du terme logiciel Open Source dans la littérature pour désigner le logiciel libre. Mais surtout, elle permet de montrer que sur la question du rapport du Libre à la gratuité, la communauté est divisée, ce qui n'est pas sans intérêt pour comprendre les différents Business Model.

Comme en français, le mot « libre » n'entraîne pas de confusion entre les notions de liberté et de gratuité, nous emploierons indifférent les termes « Open Source » et « libre ».

III. Cadre juridique

1. Introduction

Comme nous l'avons vu, les logiciels libres obéissent à des licences plus ou moins ouvertes. Dans le cas des logiciels libres, le système des droits d'auteurs a été « détourné », inversé, pour favoriser leur développement au travers de leur ouverture et

de leur garantie. Ces logiciels sont munis d'une licence qui, au lieu de limiter la diffusion ou l'usage du logiciel, protège au contraire ces droits au bénéfice de l'ensemble du public (les restreignant parfois uniquement pour certains types d'utilisations commerciales). Un inconvénient de cette très grande liberté, presque équivalente à celle du domaine public, est que ces logiciels pourraient se voir détournés en version propriétaire commerciale, en abusant de la popularité de la version libre et du travail des auteurs originaux, qui seraient alors eux-mêmes privés des améliorations apportées à leur œuvre. Pour garantir qu'un logiciel placé sous licence « libre » ne puisse d'une part être récupéré et devenir propriétaire, et d'autre part que son code source ne soit éclaté en parties libres et non-libres selon le bon vouloir des contributeurs, un cadre juridique très clair a été mis en place dès le début du développement en Open Source.

Pour comprendre sur quelles règles juridiques se base le logiciel « libre », il est nécessaire de décrire le rôle de la **Free Software Foundation**, de la **licence GNU-GPL**, ainsi que l'effet fédérateur du **projet GNU** que ces deux dernières ont amorcé. Il est à noter que si d'autres bases juridiques existent pour d'autres catégories de logiciels libres, la licence GPL est devenue la référence pour l'immense majorité des programmes en Open Source.

2. La Free Software Foundation (FSF)

La FSF a été fondée au début des années 80 par Richard M. Stallman, alors chercheur au laboratoire d'Intelligence Artificielle du MIT. Comme son nom l'indique, le but de cette fondation est de développer des logiciels libres, pouvant être légalement copiés, utilisés, modifiés et redistribués sans la moindre contrainte, les sources de ces logiciels devant rester disponibles gratuitement pour quiconque en fait la demande.

Encore une fois, il est important de comprendre que le mot Free dans Free Software Foundation ne doit pas être traduit comme gratuit mais bien comme libre. Ces logiciels peuvent tout à fait être vendus et exploités à but commercial, mais il existe toujours un moyen légal de se les procurer gratuitement.

« Lorsque nous parlons de free software, nous entendons free dans le sens de liberté, et non pas de gratuité. Notre licence est conçue pour s'assurer que vous avez la liberté de distribuer des copies des

programmes, gratuitement ou non, et que vous recevez ou pouvez obtenir le code source, que vous pouvez modifier les programmes ou en utiliser des parties dans d'autres programmes libres, en sachant que vous pouvez le faire. » Source FSF⁶

La Free Software Foundation est à l'origine de la licence GPL.

3. La licence GPL (General Public License)

La licence GPL est une licence qui spécifie les conditions de distribution de tous les logiciels GNU. La LGPL (Library General Public License) est son équivalent pour les bibliothèques de sous-programmes.

Ces licences spécifient que les logiciels GNU peuvent être copiés, modifiés et redistribués de quelque manière que ce soit, aussi longtemps que les sources sont disponibles gratuitement.

Le gros avantage des logiciels distribués selon ces conditions est que si quelqu'un désire les améliorer, tout d'abord il pourra le faire, mais également distribuer (gratuitement ou non) la nouvelle version. De ce fait, tout le monde en profitera et pourra à son tour l'améliorer. Cette méthode de développement conduit à d'excellents programmes écrits par des dizaines de personnes différentes et crée ainsi un cercle vertueux dans le développement du logiciel. Personne ne peut s'approprier un programme placé sous GPL.

Extrait de la GPL :

« Afin de protéger vos droits, nous devons faire des restrictions qui interdisent à quiconque de vous refuser ces droits ou de vous demander d'y renoncer. Ces restrictions vous imposent par conséquent certaines responsabilités si vous distribuez des copies des programmes protégés par la Licence Publique Générale ou si vous les modifiez.

Par exemple, si vous distribuez des copies d'un tel programme, gratuitement ou non, vous devez transmettre aux utilisateurs tous les droits que vous possédez. Vous devez vous assurer qu'ils reçoivent ou

⁶ <http://www.gnu.org/philosophy/selling.fr.html>

qu'ils peuvent se procurer le code source. Vous devez leur montrer cette licence afin qu'ils soient eux aussi au courant de leurs droits. »

Le cadre posé par la GPL l'a été pour permettre d'entamer le projet GNU et de garantir son maintien en tant que système basé sur le logiciel libre. Ainsi on ne peut dissocier la Free Software Foundation, la licence GPL et le projet GNU.

4. Le projet GNU

Le projet GNU (nommé ainsi par clin d'œil aux programmeurs qui apprécient la récursivité de l'acronyme : GNU = GNU is Not Unix) est un projet de la Free Software Foundation dont le but est de développer un système d'exploitation complet et entièrement libre, comprenant donc également tous les outils associés et nécessaires à son élaboration, distribué selon les conditions de la GPL. Ce système d'exploitation reprend un certain nombre de concepts de UNIX mais ce n'est pas UNIX. S'il ressemble à UNIX, c'est principalement du fait que dès ses débuts AT&T a rendu le code source de son système UNIX accessible dans les grandes universités américaines, mais sans en permettre l'utilisation. Richard Stallman a commencé le projet GNU, juste après avoir créé la FSF.

Pour le mener à bien, il a d'abord mis au point les outils indispensables à la création logicielle et les a placés sous licence GPL. La première partie a consisté à écrire un éditeur avec lequel il puisse éditer ses programmes, le célèbre GNU EMACS (« Editor MACroS »). Ensuite, il a écrit un compilateur C (le logiciel GCC) pour pouvoir compiler son système d'exploitation, c'est à dire le transformer en langage binaire adapté à la machine. Depuis lors, un certain nombre de personnes se sont jointes à lui pour écrire toutes sortes de programmes. Le système d'exploitation du projet GNU en lui-même, nommé HURD, est disponible depuis peu. Le projet GNU est aussi associé à d'autres systèmes d'exploitation, clones d'Unix et soumis aux conditions de la GPL, dont le plus médiatisé est Linux. Cependant, bien que proches techniquement et philosophiquement, ainsi que compatibles grâce aux bibliothèques GNU C «*conformes aux standards ANSI/ISO, BSD, POSIX, Single Unix, SVID et X/Open*», GNU-Hurd et GNU-Linux sont deux systèmes distincts.

En plus des principaux logiciels GNU, il existe des versions GNU de la plupart des utilitaires UNIX. Ces versions n'ont souvent rien à envier à leurs équivalents propriétaires.

IV. Bref historique

Bien avant que l'Internet et la mise en réseau progressive des ordinateurs du monde entier ne démultiplie l'importance des logiciels et ne permette le travail collaboratif à la base du logiciel libre, il existait un besoin de faire fonctionner les ordinateurs sur la base de programmes librement disponibles pour tout le monde. Ce désir de liberté provenant d'une réaction contre les logiciels propriétaires issus des supercalculateurs demande un petit rappel historique.

1. Le modèle Unix

A la fin des années 60, bien qu'IBM soit alors le plus grand vendeur d'ordinateurs généralistes, la compagnie American Telephone & Telegraph, (AT&T) détenant le monopole du téléphone aux Etats-Unis, était de taille encore plus grande et utilisait ses propres outils informatiques (matériels et logiciels) en interne. Les Bell Labs, département de la recherche d'AT&T ont donné naissance à un système d'exploitation appelé Unix. L'idée d'Unix était de créer un système d'exploitation simple, s'adaptant à toutes les échelles, pour tous les ordinateurs, des petits aux grands supercalculateurs qu'AT&T construisait pour son propre usage. Pour atteindre ce but, il fallait écrire un système d'exploitation d'un nouveau type, c'est à dire ni dans un langage machine ni dans un langage assembleur dont la forme reste liée au matériel utilisé, mais dans un langage plus expressif et généraliste. Le langage retenu était aussi une invention des Bell Labs, appelé « C ». Le langage C, alors révolutionnaire, s'est depuis généralisé et est même devenu dominant pour de nombreux types de programmation. A la fin des années 1970, le système d'exploitation Unix écrit en C a été porté sur des ordinateurs de nombreux constructeurs et de conceptions variées.

En effet, AT&T a largement distribué Unix, et en raison de la conception même du système d'exploitation (multi-plateformes), la compagnie devait effectuer cette distribution sous forme de code source C. Si le client avait des problèmes avec Unix (bug, fonctionnalité manquante), c'était à lui de plonger dans le code source afin d'y apporter les modifications nécessaires. Mais AT&T a conservé la propriété du code source et a contraint les utilisateurs à acheter des licences qui ont interdit la redistribution et la création de travaux dérivés. Les gros centres informatiques, industriels ou académiques, pouvaient se permettre d'acheter de telles licences, mais pas les individus qui en appréciaient pourtant les nombreux avantages. En même temps, les restrictions des licences interdisaient aussi à la communauté des utilisateurs qui se servaient d'Unix de l'améliorer autrement que de façon épisodique. Et comme les programmeurs à travers le monde commençaient à aspirer à une révolution de l'ordinateur personnel (et même à l'attendre), le statut « non libre » d'Unix a commencé à devenir une source de problèmes.

2. Les premiers systèmes libres

L'apparition des logiciels libres remonte apparemment à l'habitude universitaire de mettre à la disposition de toute la communauté les résultats théoriques ou expérimentaux des recherches. Cette habitude s'est logiquement étendue aux logiciels, produits très tôt en milieu universitaire vu leur fort contenu théorique. Depuis longtemps, les résultats des recherches universitaires sont utilisés en dehors de cet environnement et généralement sous forme propriétaire, en particulier dans l'industrie.

Cependant, certains auteurs universitaires de logiciels se sont progressivement convaincus que, même sans les importantes structures associées à la production industrielle et commerciale, ils étaient capables de produire des logiciels de qualité comparable et pouvant rivaliser avec leurs concurrents "professionnels.

Richard Stallman, alors employé au laboratoire d'intelligence du MIT, fut le premier à imaginer un projet de reconception et d'amélioration d'un système d'exploitation composé de vrais logiciels libres, basé sur le partage de connaissances et la coopération entre programmeurs. La raison d'être du logiciel libre serait la liberté pour tous de modifier et redistribuer de tels logiciels, avec pour seule restriction de ne pas réduire les droits de ceux à qui ils seraient redistribués. La gratuité ne faisait pas partie de

ses objectifs, bien qu'elle soit une conséquence pratique de la liberté du code source. Par ce système d'ouverture, Stallman voulait que le logiciel libre puisse devenir un projet auto-organisé, s'enrichissant de lui-même, dans lequel aucune amélioration ne serait perdue pour d'autres à travers les copyrights. Ce système a été nommé GNU, signifiant « *GNU is Not Unix* » pour marquer la différence fondamentale entre ces deux systèmes pourtant similaires sur de nombreux points. Malgré des doutes sur la conception fondamentale d'Unix (système « *pas trop mauvais* » d'après Stallman) et sur ses conditions de distribution restrictives, GNU était conçu pour bénéficier de la large (bien que non libre) distribution de sources d'Unix, connues par un grand nombre de programmeurs. Richard Stallman a commencé le projet GNU en écrivant lui-même des composants du système final qui étaient aussi conçus pour fonctionner sans modification sur les systèmes Unix existants. Le développement des outils GNU pouvait ainsi se faire directement dans l'environnement des universités et des autres centres de calcul avancé à travers le monde.

Pour mener à bien un projet d'une telle ampleur, il fallait parvenir à recruter, organiser et motiver un grand nombre de programmeurs volontaires qui construiraient tous les outils nécessaires au système d'exploitation. Pour cela, des cadres juridiques clairs ont dès le début été mis en place comme nous l'avons vu avec la Free Software Foundation et la licence GPL. Cependant ce mode de production ne s'est véritablement exprimé qu'à partir de 1991 avec le projet de Linus Torvalds, un étudiant en informatique de l'université d'Helsinki, qui a commencé le projet Linux et a vraiment dynamisé la vision et l'énergie du logiciel libre.

3. L'arrivée de Linux

Linus Torvalds a en fait commencé à adapter un outil informatique pédagogique à un usage réel. Le noyau MINIX d'Andrew Tannenbaum était une base de cours sur les systèmes d'exploitation, fournissant des exemples de solutions de base à des problèmes de base. Lentement, et d'abord sans le reconnaître, Linus Torvalds a commencé à transformer le noyau MINIX en un vrai noyau Unix pour les processeurs Intel x86, qui fonctionnent toujours sur les ordinateurs personnels de base du monde entier. Au fur et à

mesure du développement personnel de son noyau, qu'il appela Linux, il réalisa que la meilleure manière de faire fonctionner le projet était d'ajuster ses décisions liés à la conception du système afin que les composants GNU (donc libres) soient compatibles avec son noyau.

Le résultat du travail de Torvalds aboutit, en 1991, à la distribution sur l'Internet d'une esquisse de modèle fonctionnel d'un noyau libre pour un système d'exploitation sur PC semblable à Unix, entièrement compatible, et conçu de manière à pouvoir reprendre l'énorme ensemble de composants systèmes de haute qualité créés par le projet GNU de Stallman et distribués par la Free Software Foundation. Comme Torvalds avait décidé de distribuer le noyau Linux sous la Licence Générale Publique (GPL) de la Free Software Foundation, les centaines et finalement milliers de programmeurs à travers le monde qui ont décidé de contribuer par leurs efforts au développement futur du noyau étaient assurés que leurs efforts auraient pour résultat un logiciel perpétuellement libre, que personne ne pourrait transformer en produit propriétaire. Les contributeurs agissaient en sachant que leur travail serait accessible, améliorable et redistribuable. Torvalds acceptait les contributions volontiers, et grâce à son style de direction efficace le projet a maintenu son unité et l'enthousiasme des développeurs. Le développement du noyau Linux a prouvé que l'Internet rend possible le travail collaboratif d'ensembles de programmeurs bien plus grands que n'importe quel éditeur commercial ne pourrait se le permettre. Comme le rappellent de nombreuses analyses sur le mouvement Open Source, une telle échelle de collaboration entre des volontaires non payés et dispersés géographiquement, n'avait auparavant jamais eu lieu dans l'histoire humaine.

En 1994, Linux a atteint la version 1.0, représentant un noyau utilisable en production et le niveau 2.0 a été atteint en 1996. En 1998, avec le noyau en version 2.2.0 et disponible non seulement pour les machines à base de x86 (= PC équipé d'un processeur Intel) mais pour toute une variété d'autres architectures de machines, GNU/Linux (soit la combinaison du noyau Linux et des très nombreux composants du projet GNU) et Windows NT étaient les deux seuls systèmes d'exploitation du monde à gagner des parts de marché. Une évaluation interne à Microsoft ayant filtré en octobre 1998 (et ensuite reconnue comme authentique par l'entreprise...mais peut-être délibérément dans le cadre du procès anti-trust en cours) concluait que « *Linux représente un UNIX qui sort du rang, en qui on fait confiance pour des missions d'applications*

critiques et (ce qui est en partie dû au code source [sic] ouvert) et qui a une crédibilité sur le long terme qui excède celle de nombreux systèmes d'exploitation compétitifs. »⁷.

Les systèmes GNU/Linux sont maintenant utilisés à travers le monde, fonctionnant partout, faisant office de serveurs web dans des sites de commerce électronique majeurs ou en tant que clusters dédiés à l'infrastructure réseau de centres de paiement de banques. On trouve aussi GNU/Linux à des endroits surprenants, comme dans la navette spatiale ou jusqu'à récemment encore sur les serveurs de Microsoft. Les évaluations de l'industrie sur la fiabilité des systèmes Unix ont montré de manière répétée que Linux est de loin le noyau Unix le plus stable et le plus fiable, dont la fiabilité est seulement dépassée par les outils GNU eux-mêmes. GNU/Linux ne dépasse pas seulement les versions propriétaires d'Unix pour les PC dans les tests de performances, mais est renommé pour sa capacité à fonctionner, sans perturbation et sans plainte, pendant des mois sur des environnements de haut volume et de haute sollicitation sans se planter.

V. Conclusion

Dans cette première partie nous avons réalisé une présentation générale du logiciel libre. Nous l'avons définie avec précision, puis nous avons rappelé son cadre juridique et son contexte historique.

De cette présentation générale, certains points sont à retenir plus particulièrement dans le cadre de notre réflexion sur les Business Models du Logiciel libre.

Un logiciel libre se caractérise par le fait qu'il est distribué avec son code source. C'est pour cette raison que l'on peut aussi le désigner par l'appellation « logiciel Open Source ». De cette ouverture du code source découlent les quatre libertés qui caractérisent le logiciel libre : libertés d'utilisation, de compréhension du fonctionnement, de distribution et de modification. Contrairement, à l'opinion répandue, il n'y a pas, dans le logiciel libre, de notion de gratuité mais de liberté. Certes, cette liberté peut entraîner la gratuité mais pas nécessairement.

Une autre idée préconçue et fautive est que le logiciel libre n'aurait pas de légitimité juridique. Au contraire, le logiciel libre repose sur des aspects légaux très

⁷ <http://www.opensource.org/halloween/halloween1.html>

solides. Le logiciel libre n'abandonne pas le droit d'auteur, au contraire, il utilise ce droit pour assurer la pérennité de sa liberté : c'est le principe du copyleft.

Le logiciel libre implique une approche radicalement différente de la production de logiciel de celle du logiciel propriétaire. Ce dernier utilise le système de licence pour rémunérer les entreprises éditrices. Le Libre possède au contraire des licences qui par définition interdisent ce processus. Nous pouvons donc nous interroger sur les fondements économiques du logiciel libre, c'est ce que nous allons faire dans la deuxième partie.

Partie II

Economie du
logiciel libre

I. Introduction

Dans la première partie de ce mémoire sur les Business Models du logiciel libre, nous nous sommes attachés à faire une présentation générale du logiciel libre : ses aspects techniques, juridiques et historiques. Désormais nous allons nous pencher sur l'économie du logiciel libre.

Avant d'étudier les Business Models du logiciel libre il nous paraît important de comprendre ses fondements économiques. Dans cette partie nous allons chercher à voir si le Libre est viable économiquement, en particulier, s'il représente réellement une alternative sérieuse aux logiciels propriétaires.

Il existe une abondante littérature sur cette problématique, aussi allons-nous nous baser principalement sur celle-ci pour étudier l'économie du logiciel libre.

Dans un premier temps, nous allons remarquer que le logiciel est un bien économique qui possède des propriétés particulières, proche de celle de la connaissance. Dans un deuxième temps, nous allons voir que ces caractéristiques originales, permettent d'envisager une forme de production innovante, basée sur un modèle coopératif. Enfin, dans une dernière partie nous verrons que cette forme de production coopérative donne au Libre des qualités particulières.

II. Economie du logiciel

1. Aspects généraux

a. Introduction

Dans la première partie nous avons vu les aspects techniques et juridiques de l'industrie logicielle. Dans cette partie nous allons faire le point sur la littérature économique concernant l'économie du logiciel.

Nicolas Jullien (2001) nous explique qu'un logiciel peut apparaître dans un premier temps à un bien informationnel (au sens d'Arrow (1962), est en fait, une forme d'écriture (Lévy, 1992), car il nécessite un langage d'écriture et il permet de produire un texte actif (un «texte qui agit», Callon (1991)). Ce texte actif représente pour Zimmermann (1995) «*l'expression, dans un langage formalisé, d'un schéma de calcul constitué de l'articulation d'algorithmes*», qui permet de «mettre en oeuvre un processus sans requérir de la part de l'utilisateur, une maîtrise intellectuelle effective des composantes de ce processus». À cause de cela, on peut dire que programmer, créer du logiciel est une activité de codification de connaissance, à la fois connaissance d'un besoin, d'une tâche à automatiser ou à faciliter, et connaissance de la façon de réaliser cette tâche. L'analyse de l'économie du logiciel s'appuie donc sur les travaux qui ont étudié la façon dont sont produites les connaissances.

Même si Mowery (1996) note que la production non marchande a toujours représenté une part très importante de la production globale de code source, aujourd'hui la production industrielle, marchande de logiciel est largement dominante et c'est celle qui a été la plus étudiée en économie. Comprendre comment est produit le logiciel, c'est d'abord comprendre comment fonctionne cette industrie

b. Economie du logiciel propriétaire commercial

Nicolas Jullien (2001) fait remarquer qu'il n'existe qu'un petit nombre de travaux sur le sujet du logiciel. Mowery (1996), l'un des rares auteurs ayant travaillé sur le sujet, écrit d'ailleurs que «l'industrie du logiciel a peu attiré l'attention de manière surprenante, étant donné sa taille, sa croissance rapide et son importance de plus en plus grande dans les industries de haute technologie». Cela tient sans doute d'abord à ses caractéristiques, qui en font un produit particulier. En effet, il se distingue de certains biens intangibles comme la musique ou l'oeuvre littéraire, qui n'ont pas d'objectifs utilitaristes. On pourrait le rapprocher des plans, des modèles, qui sont aussi des prototypes, mais contrairement à eux, le logiciel décrit un processus et permet de l'exécuter (Brousseau 1993). Il se distingue aussi des services, puisque c'est un produit qui codifie une connaissance, une capacité technique, qui décrit un processus, sans que le producteur de cette capacité

technique ait besoin d'intervenir une fois le transfert effectué. Une fois que cette connaissance est codifiée, on dispose d'un outil, dont il faut apprendre l'utilisation (il faut comprendre quelles tâches sont réalisables avec l'outil et la façon dont elles peuvent être réalisées). Mais il peut être produit et distribué quasiment sans coût car c'est un bien intangible, et qui est de plus en plus facilement reproductible.

Nicolas jullien (2001) ajoute que le problème que pose la production du logiciel est celui du financement de la production-codification de ce bien «public», tel qu'à pu le définir Samuelson (1954, 1955). Cette production se fait traditionnellement soit grâce à un financement public, soit par la construction d'un système juridique qui permet de créer un marché en rendant le bien exclusif et donc en permettant de rémunérer les producteurs par la facturation d'un droit d'usage (Coase 1959).

A cause des spécificités du bien «logiciel», cette industrie possède quatre caractéristiques originales (Richardson 1997), qui expliquent la constitution de monopoles, mais aussi le dynamisme et le renouvellement continu des entreprises productrices de logiciels. Conséquence directe du fait que ce soit un bien «public», le coût de développement d'un logiciel n'est pas affecté par l'ampleur de la population des utilisateurs. En outre, le coût d'extension marginal de cette population est nul ou réduit à un montant négligeable par rapport au coût de développement. Le rythme de l'innovation est élevé, ce qui réduit la durée de vie des produits. Ces deux caractéristiques engendrent une concurrence forte, parfois agressive au niveau des prix, pour imposer sa solution et pouvoir jouir d'une rente de monopole. Les deux autres spécificités renforcent parfois ce type de concurrence, mais les limitant aussi: il existe des «effets de réseau», dans la mesure où un logiciel «n'est d'aucune utilité en lui-même, mais seulement quand il est mis en oeuvre conjointement avec d'autres produits complémentaires au sein d'un système». Les entreprises qui possèdent un logiciel sont alors tentées de développer les logiciels complémentaires. Mais, en même temps, elles peuvent difficilement couvrir l'ensemble du spectre de la demande et de nouvelles entreprises se créent sans cesse pour répondre à de nouveaux besoins. La conséquence en est que les standards jouent un rôle très important car ils permettent de mettre en oeuvre conjointement des produits complémentaires. Là encore, la définition des standards favorise l'entreprise qui contrôle leur évolution, elle peut plus facilement anticiper sur leur évolution et garantir l'interopérabilité des programmes complémentaires. D'un autre côté, le fait qu'elle ne

puisse pas répondre à l'ensemble des demandes l'oblige à rendre publiques leurs caractéristiques.

2. Les bonnes propriétés

Outre les caractéristiques générales du logiciel d'un point de vue économique que nous venons de développer. Foray et Zimmermann (2002) insistent sur certaines propriétés propres aux logiciels :

« - premièrement, un logiciel constitue un objet scientifique ou technologique complexe qui est donc caractérisé par des processus d'apprentissage quasi-illimités: un système composé de milliers de développeurs travaillant sur le même logiciel durant une longue période restera longtemps, presque infiniment, dans une phase de rendement croissant - ce qui ne serait pas le cas d'un système de milliers d'ingénieurs travaillant à l'amélioration d'une brouette!

- Deuxièmement, un logiciel est exprimé sous la forme d'un ensemble d'instructions codifiées qui circulent parfaitement sur le réseau électronique. Ainsi, la circulation des améliorations apportées est rapide, parfaite et son coût marginal est nul. Ceci accroît l'efficacité globale du processus de recherche collective.

- Troisièmement, les logiciels appartiennent à une certaine classe de technologie ayant la propriété particulière de réduire, voir même d'annuler la distance entre producteurs et consommateurs (Quah, 1999). Les millions d'utilisateurs qui révèlent les problèmes sont pour une part les développeurs qui proposeront les solutions. Le va-et-vient entre identification des problèmes et formulation des solutions est donc quasi-instantané. »

Ces caractéristiques du logiciel comme un mode d'organisation de la création de logiciels qui fonctionnerait de manière ouverte, coopérative et grâce à Internet, pourraient se révéler particulièrement efficace.

Ce mode de production ouvert aurait des caractéristiques similaires aux “ bonnes propriétés ” de la distribution de la connaissance et des systèmes de savoir ouvert: seule la circulation rapide et élargie des savoirs permet de bénéficier du potentiel unique d'un très grand nombre d'individus compétents. D'un point de vue concret, une distribution

rapide de la connaissance facilite la coordination entre les agents, elle diminue les risques de duplication entre projets de recherche et surtout, en propageant la connaissance au sein d'une population diverse de chercheurs et d'entrepreneurs, elle accroît la probabilité de découvertes et d'inventions ultérieures, en même temps qu'elle diminue le risque que cette connaissance soit détenue par des agents incapables d'en exploiter les potentialités (David et Foray, 1995).

Ce mode de production ouvert, que permettent certaines propriétés du logiciel, est un des fondements majeurs de l'économie du logiciel libre que nous allons désormais étudier.

III. Modèle de production du Libre

1. Introduction

Dans cette partie nous allons voir que ce qui caractérise le logiciel libre d'un point de vue économique, c'est son mode de production originale qui repose sur les « bonnes propriétés » du logiciel que nous venons d'étudier.

Pour comprendre le fonctionnement et les implications de cette forme originale de production, nous allons analyser en détail deux textes qui jouent un rôle particulier dans la réflexion sur le modèle de développement du libre. Ces textes ont été écrits par les deux principaux « gourous » du mouvement Open Source : Eric S. Raymond et Linus Torvalds.

2. La cathédrale et le Bazar, Eric S. Raymond 1998

a. Introduction

Le texte la « cathédrale et le bazar » est sans aucun le texte de référence du monde du logiciel libre. Il est le premier à avoir exposé de manière claire les différences entre la conception d'un logiciel propriétaire (la cathédrale) et celle d'un logiciel libre (le bazar)

Ce texte a été écrit par Raymond, qui est un hacker, un de ces « bidouilleurs » qui contribua au développement des logiciels libres. Raymond a été impliqué dès les années 80 dans le développement de UNIX, ancêtre de linux ; il a également contribué à GNU ainsi qu'à de nombreux autres projets Open Source.

Dans ce texte, Raymond montre la mise en place d'un mode de développement Open Source, à partir de sa propre expérience dans le développement de la messagerie « Fetchmail ». Ce texte s'adresse dans un premier temps aux hackers pour leur prouver qu'un modèle de développement Open Source est efficace et leur donner ainsi l'envie de participer. Dans un second temps, on se rend bien compte, dans les apports des différentes versions, qu'il s'agit aussi de convaincre certaines sociétés éditrices de logiciels, du bienfait pour elles de passer en mode de développement Open Source. Ce faisant, il apporte à l'économiste des concepts fondamentaux pour mieux comprendre le développement du logiciel libre.

b. La cathédrale et le bazar

Les deux premiers concepts, que l'on va retrouver en permanence dans la littérature qui analyse le phénomène du libre, sont ceux de cathédrale et de bazar.

La cathédrale désigne le mode de développement habituel et fermé des logiciels. Les programmeurs travaillent dans une firme qui coordonne leurs travaux. Le code source des logiciels qu'ils produisent est fermé et ils ne les distribuent que quand ils sont parfaitement opérationnels ou du moins devraient l'être.

La cathédrale symbolise l'ordre, le fonctionnement hiérarchique qui caractérise ce modèle. A contrario, le concept de bazar s'applique au mode de développement qui caractérise le modèle open source.

c. Fonctionnement du bazar

Dans cette étude, nous allons laisser de côté les aspects techniques et l'historique du mouvement de « Fetchmail » pour nous concentrer sur les renseignements que retire Raymond de cette expérience. L'enseignement principal est une comparaison du concept de bazar avec le modèle d'incitation du monde scientifique.

Dans le bazar, le développeur va travailler sur quelque chose qui l'intéresse. En effet, il développe d'abord pour lui, pour améliorer ou créer un programme qui lui convient. Il sera plus efficace qu'un développeur du monde propriétaire à qui l'on a assigné une tâche. Si un logiciel est Open Source cela signifie que vous pouvez le modifier à condition de laisser la nouvelle version modifiée en libre accès à tous. Ce qui

est donc intéressant, c'est de ne pas hésiter à reprendre le travail d'un autre au lieu de tout refaire soi-même.

D'ailleurs Raymond souligne l'importance des autres dans le travail Open Source. D'abord il est utile de trouver un maximum d'utilisateurs du logiciel, car ceux-ci voudront l'améliorer et ainsi participeront à son développement. De plus, grâce à Internet, ces collaborations seront renforcées. Raymond insiste beaucoup sur le fait que les utilisateurs sont les mêmes que ceux qui développent le logiciel. Pour cette raison il conseille aussi des mises à jour fréquentes et de distribuer même des versions non opérationnelles. Ainsi, la communauté qui s'est créée autour d'un projet pourra tout de suite la tester, soulever les bugs, voire même les réparer. Tout ce phénomène qui peut sembler créer du désordre, en fait arrive à une solution efficace. Raymond prouve par exemple que les logiciels libres sont moins « bugés » que leurs homologues fermés dans la mesure où de nombreux utilisateurs les ont testés souvent au cours de leurs développements ; la probabilité de passer à côté d'un bug moyen est bien plus faible que dans la cathédrale. Ainsi dans les logiciels libres il existe une version « n » parfaitement stable et une version « n+1 » plus novatrice en voie de stabilisation.

d. Les facteurs de la réussite d'un projet Open Source

Au delà de cette description du bazar, ce qui est très intéressant dans le texte de Raymond, est l'examen qu'il mène sur les pré-requis nécessaires au fonctionnement d'un projet Open Source.

Il est nécessaire d'avoir un point de départ : un logiciel minimaliste qui fonctionne. Il faut que ce logiciel soit programmé proprement pour partir sur de bonnes bases. Il faut également convaincre les participants futurs des potentialités du projet. D'ailleurs la personnalité du concepteur du projet est importante.

Il est nécessaire qu'il sache garder les bonnes idées des autres, tout en conservant une cohésion à l'ensemble. De façon générale, le coordinateur du projet doit être doué pour les relations humaines.

Dans l'ensemble, un projet Open Source est aussi une gestion de projet originale. D'abord il y a le rôle important joué par Internet pour coordonner les acteurs. Mais surtout, ce qui est fondamental dans la compréhension, ce sont les éléments qui motivent

les développeurs à participer. C'est d'abord l'amélioration d'un logiciel qui les intéresse. Pour Raymond, ce qui motive les développeurs, c'est leur satisfaction personnelle et leur réputation au sein de la communauté des autres « bidouilleurs ». Les morceaux de programmes sont signés par leurs auteurs. Ainsi un développeur particulièrement efficace pourra se faire remarquer par la communauté.

Raymond conclut que la culture du logiciel libre pourrait à terme triompher non pas parce qu'elle a une meilleure morale, mais tout simplement parce qu'elle est plus efficace économiquement.

L'intérêt majeur de ce texte est surtout son impact historique dans la réflexion sur l'économie du logiciel libre, en particulier dans l'analyse du fonctionnement et des mécanismes d'incitations qui expliquent le fonctionnement du bazar.

Le texte suivant va nous permettre d'approfondir le concept de bazar et en particulier d'affiner la problématique à laquelle une société est confrontée quand elle décide de passer d'un mode de développement fermé à un mode de développement ouvert.

3. Pourquoi le modèle Open Source est sensé, Linus Torwalds, 2001

a. Introduction

Le deuxième article que nous allons étudier s'intitule « Pourquoi le modèle open source est sensé ». Il s'agit d'un chapitre du livre « Il était une fois Linux », écrit par Linus Torwalds en collaboration avec un journaliste spécialisé en informatique, David Diamond, en 2001. Linus Torwalds est le père de Linux, le plus célèbre des logiciels open source et véritable figure emblématique de ce mouvement. Le point de vue d'un membre aussi éminent et influent de la communauté du Libre que Linus Torwalds, est intéressant et enrichissant. Ce texte est conçu comme un prolongement de la réflexion entamé par Raymond (1998).

b. Rappel des caractéristiques du bazar

Dans son article, Linus Torwalds reprend les principes qui caractérisent le bazar de Raymond et y apporte un éclairage différent. Il apporte surtout des précisions, du fait de sa place originale au sein du mouvement du logiciel libre. Il prend aussi en compte les évolutions que ce mouvement a connues entre 1988 et 2001.

La première caractéristique de la philosophie Open Source, rappelée par Linus Torwalds, est que dans ce modèle «*n'importe qui peut se joindre à un projet de développement ou de commercialisation*». Cette caractéristique peut s'avérer un vrai facteur de réussite puisque c'est ainsi que IBM a réussi à imposer son PC compatible comme standard de la micro-informatique, faisant disparaître ses concurrents : Atari, Commodore ou même Apple.

Linus Torwalds insiste beaucoup sur l'importance de la communauté, surtout dans la capacité d'un groupe à mieux trouver les bugs que ne pourrait le faire une personne seule. Il remarque que cette caractéristique fondamentale du fonctionnement du logiciel libre pourrait facilement être adoptée dans d'autres secteurs d'activités complexes, grâce en particulier au développement des technologies de l'information. Pour revenir dans le domaine des logiciels, Linus Torwalds constate que le modèle du logiciel libre permet d'atteindre les meilleurs standards de qualité.

c. Le modèle de la recherche scientifique et les effets secondaires insoupçonnées de la « philosophie Open Source »

Comme pour Raymond, Linus Torwalds montre que l'apprentissage et la reconnaissance de la communauté sont des facteurs de motivation importants pour les développeurs. Mais il ajoute un autre aspect, qui est le côté passion et militantisme, qui anime les hackers. De plus il fait un rapprochement intéressant entre les moteurs de la motivation dans la recherche scientifique et le monde du logiciel libre. Ce rapprochement a été développé par Kelty (2001) et abordé par Foray et Zimmermann (2002). Du fait de la montée en puissance du logiciel libre, de nombreuses sociétés se créent dans ce domaine et embauchent les meilleurs développeurs, ce qui ajoute une source de motivation nouvelle pour participer à la communauté.

La diffusion d'un logiciel libre a des effets secondaires insoupçonnés. Cela permet aux entreprises secondaires d'adapter le logiciel à leurs besoins ou aux nouveaux besoins du marché. Le logiciel va évoluer et va avoir de nombreuses utilisations innovantes.

Ainsi pour Linus Torwalds : « En évitant de contrôler la diffusion de la technologie, vous ne limitez pas les possibilités d'emploi ».

d. Quand une entreprise est l'initiatrice d'un projet Open Source

Linus Torwalds se penche aussi sur les difficultés pour une entreprise de passer d'un modèle fermé à un modèle ouvert. Raymond s'intéressait aux problèmes du rôle des individus (développeurs, utilisateurs, concepteurs de projets...) dans le cadre d'une communauté créée autour d'un projet de logiciel Open Source. Ici, Linus Torwalds aborde la problématique de l'entreprise qui souhaiterait développer un logiciel Open Source.

D'abord il est important de prendre en compte la peur du changement. En restant dans un modèle fermé, l'entreprise peut mieux prévoir son évolution et donc s'adapter aux changements. En adoptant un modèle ouvert, l'entreprise a des chances de beaucoup mieux réussir, mais ce succès est plus imprévisible. Le dilemme est entre une réussite plus complète mais plus incertaine ou une réussite moindre mais plus prévisible.

Pour une entreprise qui veut basculer d'un logiciel fermé à un logiciel ouvert, il existe un autre problème majeur : « L'entreprise a accumulé beaucoup de connaissances et de savoir faire qui font partie de son capital ». Cette connaissance interne reste confidentielle, ce qui est une barrière vers l'extérieur.

Linus Torwalds insiste donc sur la nécessité d'associer l'extérieur, c'est-à-dire les futurs développeurs, aux prises de décisions. Cette démarche a le double avantage de créer une dynamique positive entre l'extérieur et l'entreprise et d'apporter des idées importantes et intéressantes de l'extérieur vers l'entreprise.

e. Les conséquences pour une entreprise de s'ouvrir

Cependant cette contribution extérieure n'est pas sans conséquences au sein même de l'entreprise. Il faut gérer les problèmes de blessures d'amour propre des salariés. En effet, le problème se pose quand un contributeur externe à la société apporte un travail qui annule et remplace celui d'une personne interne à l'entreprise et que cela transpire à l'extérieur. Linus Torwalds en tire comme conclusion que cela devient impossible de se cacher derrière ses supérieurs hiérarchiques. Afin de concilier la nécessité d'intégrer

l'extérieur et la gestion des problèmes d'amour propre, il en déduit le besoin de repenser l'organisation de l'entreprise. Il confère un rôle très important au responsable du projet : ce dernier doit maintenir les objectifs de l'entreprise par rapport au projet, mais aussi adapter celui-ci aux besoins réels. On retrouve dans ce texte les descriptions des qualités du concepteur du projet décrites chez Raymond: l'expertise technique, une vue d'ensemble. Linus Torwalds ajoute des champs de compétence complémentaires à son responsable de projet : la capacité à faire l'interface entre l'entreprise et l'extérieur, une bonne prise en compte des objectifs de l'entreprise sans être trop dépendant de celle-ci pour conserver sa légitimité. Pour Linus Torwalds ce « responsable de projet » est la véritable clef de voûte d'un projet « Open Source » en particulier quand il est lancé par une entreprise.

Linus Torwalds conclut comme Raymond qu'il y a un ordre de crédibilité économique au modèle Open Source, mais il convient aussi que ce modèle « *n'est pas adapté à tous les projets, à toutes les personnalités, à toutes les entreprises* ».

f. Intérêts et limites du texte

Certes ce texte connaît certaines limites ; il ne s'agit pas d'un travail académique et nous pourrions, de bonne foi, remettre en cause l'objectivité d'un auteur aussi impliqué dans le monde de l'open source que Linus Torwalds. Cependant, ce texte ne manque pas d'intérêt, d'abord par la personnalité de son auteur, qui du fait de son rôle dans le monde du logiciel libre, est un observateur privilégié de l'évolution de ce dernier. Ce texte complète très bien celui de Raymond en ayant plus de recul sur le développement du modèle Open Source, puisqu'il a été écrit en 2001. Surtout, il a affiné l'analyse de Raymond en se penchant sur les problématiques du bazar quand une entreprise en est à l'origine. Cet aspect n'est pas négligeable dans la perspective d'une réflexion sur les Business Model des entreprises qui gravitent autour du libre.

4. Conclusion

L'analyse approfondie de ces deux textes fondamentaux nous a permis de mieux comprendre le mode de production qui caractérise le logiciel libre. Jullien (2002) parle de renouveau du modèle coopératif pour le désigner.

Si d'un point de vue technique le logiciel libre est caractérisé par une ouverture de son code source, si d'un point de vue juridique il est caractérisé par un détournement du droit d'auteur, pour garantir sa liberté (copyleft), d'un point de vue économique le logiciel libre est caractérisé par sa forme de production coopérative.

L'étude de ces textes nous a aussi montré que cette forme de production coopérative a été rendue possible grâce aux bonnes propriétés du logiciel libre (Foray, Zimmermann, 2002), à l'utilisation d'Internet et à des modes d'incitations performants.

L'incitation à participer efficacement à un projet de logiciel libre est comparable aux modes d'incitations que l'on trouve dans le monde scientifique. Ce mode d'incitation fonctionne par la reconnaissance de son travail par ces pairs : c'est ce que Lerner et Tirole (2000) définissent comme « *the ego gratification incentive* ». Ce mécanisme d'incitation est complété par le « *career concern incentive* » (Lerner et Tirole, 2000). La participation à un projet Open Source permet de signaler la qualité de ses compétences à un employeur potentiel.

L'étude de ses deux textes nous a aussi permis de comprendre, que le mouvement du logiciel libre n'était pas simplement un mode de production de logiciel original, mais aussi un système de production particulièrement efficace. Le mode de production des logiciels libres est la source même de leurs qualités.

La loi de Linus stipule qu'« *Étant donné un ensemble de bêta-testeurs et de co-développeurs suffisamment grand, chaque problème sera rapidement isolé, et sa solution semblera évidente à quelqu'un* ». Cette loi explique la grande qualité et la stabilité générale des logiciels libres.

Nous allons donc désormais chercher à voir qu'elles sont les principaux avantages que ce mode de production coopératif donne aux logiciels libres sur leurs concurrents propriétaires.

IV. Les avantages du logiciel libre

1. Introduction

Si la part de marché des logiciels libres est en constante augmentation et si certains logiciels libres comme Apache ou Linux connaissent des succès retentissants. Il n'en demeure pas moins que la forme dominante de logiciel est celle du logiciel propriétaire.

Cet état de fait a plusieurs causes. D'abord la jeunesse du Libre qui est un mouvement qui a pris son essor depuis moins de 10 ans, explique en partie sa faible part de marché dans l'industrie logicielle. Il existe, en outre, d'autres facteurs explicatifs.

Du point de vue du producteur, c'est à dire de l'éditeur de logiciel, le logiciel propriétaire représente toujours un mode de production intéressant. D'abord parce que nous sommes dans une trajectoire technologique qui est celle du logiciel propriétaire. Le logiciel propriétaire a fait les preuves de son efficacité, et reste le modèle dominant, donc beaucoup d'éditeurs préfèrent continuer de suivre cette voie. De plus du fait des caractéristiques économiques du logiciel, certaines entreprises ont réussi à se créer des rentes de monopole, c'est le cas de Microsoft par exemple. Ces entreprises n'ont aucun intérêt pour l'instant à abandonner un modèle économique qui leur est si profitable.

Du point de vue de nombreux utilisateurs, le logiciel propriétaire paraît plus attractif que le logiciel libre. Ce point de vue est basé sur des raisons psychologiques et objectives. En effet, le Libre est une nouveauté et est encore mal connu de nombreux utilisateurs. De plus la nouveauté effraie surtout quand elle consiste en un changement de trajectoire technologique aussi radicale. Mais il existe aussi de raisons objectives pour expliquer ce refus de passer au Libre. Passer au libre signifie changer complètement son système logiciel, ce qui a un coût humain très élevé pour une entreprise. De plus, les logiciels propriétaires sont souvent de très bonnes qualités, alors pourquoi changer une équipe qui gagne.

A ceci, s'ajoute les faiblesses du Libre. De nombreux logiciels propriétaires n'ont pas de concurrents libres, soit qu'il n'existe aucun projet de logiciel libre comparable, soit que ce projet n'est pas encore avancé. C'est le cas en particulier dans l'univers des jeux informatiques, où il existe très peu de jeux en logiciel libre. De plus, un autre handicap pour le logiciel libre est « l'effet de club » qu'engendre l'utilisation d'un logiciel et en particulier la mise en place de standard. Par exemple, même s'il existait un concurrent plus performant que Word comme traitement de texte, il aurait beaucoup de

mal à s'imposer. En effet, les premiers adoptants, ne pourrait plus échanger facilement des textes avec l'immense masse de utilisateurs restés fidèle à Word.

Il existe donc de nombreux facteurs qui pénalisent le logiciel libre, mais celui ci a aussi de nombreux atouts pour continuer d'accroître son importance vis-à-vis du logiciel propriétaire. Nous allons désormais voir les éléments qui font la force du logiciel libre, en particulier si l'utilisateur du logiciel est une entreprise.

2. Les avantages

a. Introduction

Le mode de développement coopératif des logiciels libres semble donc être particulièrement compétitif, malgré son apparente anarchie (principe du Bazar). Vis à vis des logiciels au développement propriétaire, on peut retenir les caractéristiques techniques suivantes dans la majeure partie des cas : des fonctionnalités avancées, une rentabilité maximum des efforts fournis et du temps investi, une amélioration « darwinienne » aboutissant à une efficacité maximum, une très grande fiabilité et un minimum de bugs, un respect supérieur des standards, une garantie de fonctionnement optimale si le suivi est maîtrisé, une liberté de choix supérieure pour les utilisateurs, et finalement la pérennité des solutions choisies.

Voyons ces différents points plus en détail :

b. Fonctionnalité

Les logiciels libres sont réalisés par des personnes passionnées par un sujet donné ou par des fonctions particulières. Par conséquent, ils disposent naturellement des fonctionnalités les plus avancées dans leurs domaines respectifs, tandis que les logiciels propriétaires ont davantage tendance à faire évoluer des techniques plus anciennes. Par exemple, la version de Windows Millénium sortie en 2000 était encore basée sur Ms-Dos qui date de 1981.

D'autre part, les logiciels libres existants peuvent servir de base fiable pour les personnes et les sociétés qui désirent ajouter des fonctionnalités spécifiques non encore

couvertes. Ils peuvent recevoir de nouvelles fonctionnalités qui n'auraient jamais été introduites dans un logiciel fermé ou propriétaire vue la spécificité de la demande.

c. Rentabilité

Cette possibilité d'ajouter de nouvelles fonctions permet de gagner énormément de temps et de ressources, puisque ceux qui veulent ajouter une nouvelle fonctionnalité peuvent réutiliser tous les codes sources existants. La déperdition d'énergie dans le développement est minimisée par les mises à jour fréquentes des améliorations. Il est en effet plus simple de partir d'un logiciel libre existant et de lui ajouter la fonctionnalité recherchée que de repartir à zéro et d'écrire un logiciel complet. Ceci a une incidence importante sur leur rentabilité par rapport aux produits commerciaux.

d. Efficacité

Par l'ouverture des sources et la possibilité de les modifier, les logiciels libres permettent la contribution de tout volontaire. Ces contributions portent parfois sur des petites parties du logiciel, et se font par des personnes différentes dans le monde entier et sans autre rapport que la base de source commune. Ceci permet l'exploration de différentes solutions techniques et la meilleure est généralement retenue, tandis que l'industrie logicielle ne peut pas se permettre une telle recherche. Sur le long terme, grâce à la sélection naturelle des solutions techniques, les logiciels libres se révèlent être les plus performants et les plus efficaces en permettant aux utilisateurs de faire du programme ce qu'ils en attendent réellement.

e. Fiabilité

Toujours grâce à l'ouverture des sources, tout utilisateur sachant le faire, a la possibilité de corriger les erreurs éventuelles qu'il peut détecter. Quasiment aucune erreur ne peut donc passer au travers de ce filtrage continu, auquel tous les utilisateurs actifs participent. Les logiciels libres atteignent donc une très grande fiabilité en peu de temps.

f. Compatibilité avec les standards

Les développeurs de logiciels libres favorisent le respect des standards. En effet, seuls les standards garantissent une interopérabilité parfaite avec les autres logiciels. Personne n'a intérêt, dans le monde du libre, à utiliser des protocoles incompatibles ou des formats de fichiers non standards, puisque les sources sont ouvertes et qu'il est impossible d'utiliser les techniques de rétention d'informations classiques dans le but de gagner des parts de marché. Les logiciels libres manipulent donc leurs données sous des formats standard, qui permettent de les récupérer et de les traiter avec d'autres logiciels de manière fiable et à moindre coût.

g. Garantie de fonctionnement

Professionnellement, la possibilité de modifier les sources garantit que les logiciels vont fonctionner de toute façon. En effet, avec les logiciels propriétaires, les utilisateurs sont absolument dépendants des sociétés éditrices en cas de problème. Les contrats de service classiques sont non seulement chers, mais souvent inefficaces, car la correction d'un bug passe souvent par l'attente de la version suivante (et de son achat).

Au contraire, les logiciels libres sont plus réactifs et permettent une correction immédiate, mais ils permettent également aux utilisateurs de choisir la solution à leur problème. Ils peuvent le résoudre eux-mêmes s'ils en ont les moyens, ou sinon louer les services d'une société spécialisée qui assure alors un fonctionnement optimal. Dans les deux cas, les utilisateurs ont l'assurance du bon fonctionnement de leurs logiciels.

h. Garantie de liberté

La disponibilité des sources garantit en permanence la liberté de tout utilisateur. Il n'est pas possible, dans un logiciel en Open Source, d'inclure des fonctionnalités cachées dans le but de restreindre les libertés individuelles ou de collecter des informations sur les utilisateurs (ce que l'on appelle le « spyware »).

Du fait qu'ils respectent les standards, les logiciels libres n'utilisent pas des formats de fichiers non documentés (comme par exemple les différentes versions de Microsoft Word) ou des protocoles de communication propriétaires. Ils garantissent donc la libre circulation des informations et la liberté d'expression de chacun quel que soit son équipement.

De plus, les logiciels libres proposent souvent une ou plusieurs alternatives aux autres logiciels, garantissant ainsi la liberté de choix à leurs utilisateurs.

i. Pérennité

La disponibilité des sources garantit aux utilisateurs la pérennité des logiciels qu'ils utilisent. L'abandon du support du logiciel par la société éditrice n'est donc pas à craindre.

3. La problématique des coûts

Si comme on l'a vu dans la première partie il faut bien traduire le préfixe free dans freesoftware par libre et non par gratuit, il n'empêche que la notion de liberté implique une certaine gratuité. Certes il est tout à fait légal de vendre un logiciel libre, mais en même temps il n'est pas du tout nécessaire de l'acheter pour l'utiliser, c'est une conséquence des libertés 0 et 2 (La liberté d'utilisation et la liberté de redistribuer des copies) qui caractérisent le logiciel libre.

Un raisonnement simpliste pourrait donc laisser à penser que le coût total d'appropriation (TCO : *Total Cost of Ownership*) d'une solution basée sur les logiciels libres seraient inférieur au TCO d'une solution propriétaire. De grandes sociétés éditrices de logiciels libres, en particulier Microsoft, ont eu beau jeu de faire remarquer que le coût d'achat des licences logicielles était négligeable par rapport aux autres dépenses qu'engendrait un changement de système informatique. Cet argument est parfaitement recevable, cependant Microsoft va plus loin en affirmant que le TCO d'une solution propriétaire est inférieur à celui d'une solution libre.

Même s'il est très difficile de chiffrer exactement un TCO, certaines études récentes tendraient plutôt à démontrer que, du fait des nombreux avantages des logiciels libres, le coût d'appropriation d'une solution libre serait en fait moins onéreux qu'une solution propriétaire.

Dans son rapport remis en juin 2001 au gouvernement le député du Tarn, Thierry Carcenac, réalise un tableau récapitulatif (fig3) sur les conséquences en terme de coût de l'adoption d'une solution basée sur des logiciels libres par rapport à une solution basée sur des logiciels propriétaires.

Installation	Plus onéreuse	Les multiples variantes existant des logiciels libres et la multiplicité des solutions peuvent rendre les choix d'installation plus complexe. Les outils d'installation sont souvent plus complexes que pour les logiciels propriétaires.
Insertion au sein du SI	Moins onéreuse	Sauf dans le cas où le SI en question est mono-source (ce qui pose des problèmes de relations fournisseurs et de sécurité), le respect des standards ouverts garanti par les logiciels libres assure un coût d'insertion au sein du SI moins important.
Matériel	Moins onéreux	Les logiciels libres semblent actuellement pouvoir se contenter de configurations matérielles plus limitées que leurs équivalents propriétaires - parfois au prix de fonctionnalités plus rustiques mais standardisées.
Licences	Moins onéreuses	Par construction, le coût de licence est à peu près nul.
Sécurisation	Moins onéreuse	Le mode de développement libre assure mieux de la sécurité des développements.
Formation	Moins onéreuse sauf coûts de transition	Les logiciels libres sont bien implantés dans le domaine universitaire et de la recherche, ce qui en fait des systèmes de choix pour la formation initiale et continue. Cependant, les informaticiens formés à des systèmes propriétaires doivent souvent être entièrement reformés, ce qui renchérit d'autant le coût de transition.
Assistance	Plus onéreuse	L'offre d'assistance en matière de logiciels libres est en cours de constitution, ce qui la rend aujourd'hui plus onéreuse que sa contrepartie propriétaire, soit directement, soit indirectement (à travers la recherche de prestataires).
Evolution	Moins onéreuse	Le respect des standards facilite la séparation entre briques logicielles que l'on peut faire évoluer indépendamment. La coopération répartie favorise le développement de programmes largement autonomes et spécialisés qui peuvent ensuite évoluer indépendamment.
Archivage	Moins onéreux	Le respect des standards ouverts et la proximité avec la recherche assurent du respect des formats d'archivage.

Fig 3. TCO du logiciel libre par rapport à une solution propriétaire
Source Rapport Carcenac⁸

Le rapport de Thierry Carcenac confirme bien que du fait des qualités conférées par sa forme de production coopérative, le coût total d'acquisition est moindre pour une solution de logiciel libre.

Une étude récente réalisée par « Cypersource »⁹, une société australienne de consulting informatique (logiciel libre et logiciel propriétaire) a comparé le TCO d'une solution Linux et Windows. Cette étude a été réalisée sur la base d'une société comprenant 250 utilisateurs et sur une durée de 3 ans. Les résultats (fig 4) confirment

⁸ <http://www.internet.gouv.fr/francais/textesref/rapcarcenac/sommaire.htm>

⁹ <http://www.cyber.com.au>

l'analyse de Thierry Carcenac, puisque selon les configurations hardware, on arrive à un TCO inférieur de 25% à 34% pour la solution Linux

	Microsoft Solution (TCO Over 3 Years)	Linux/Open Source Solution (TCO Over 3 Years)	Savings Achieved by Using Linux (Over 3 Years)	Percentage Saved (Over 3 Years)
Existing Hardware & Infrastructure is used	\$733,973	\$482,580	\$251,393	34.26%
New Hardware & Infrastructure is purchased	\$1,042,110	\$790,717	\$251,393	24.69%

Fig 4. TCO Windows/Linux
Source Cybersource¹⁰

V. Conclusion

Dans cette partie nous nous sommes attachés à comprendre l'économie du logiciel libre. Nous nous sommes donc rendus compte que ce qui faisait l'originalité du logiciel libre c'est son mode de production coopératif grâce à Internet et aux caractéristiques propres aux logiciels. Ce mode de production aussi appelé Bazar, repose sur des mécanismes d'incitations originaux car ils ne sont pas basés sur une rémunération financière directe, mais par une reconnaissance par les pairs et une signalisation des compétences sur le marché de l'emploi.

Du fait, en particulier, de cette production coopérative, il s'avère que le logiciel libre possède de nombreux avantages sur le logiciel propriétaire même s'il doit encore payer le prix de la jeunesse de ce concept. Ces avantages montrent que des Business Model basés sur le logiciel libre ne sont pas construits autour d'un projet utopique mais bien autour d'une logique qui permet de produire des logiciels de très haute qualité.

Cette partie nous apporte un autre élément important dans le cadre d'une réflexion sur le logiciel libre, il nous montre que le Bazar est capable de produire des logiciels de qualité à très faibles coûts puisque les mécanismes d'incitations des développeurs ne sont pas basés sur une rémunération.

¹⁰ http://www.cyber.com.au/cyber/about/linux_vs_windows_tco_comparison.pdf

Partie III

Typologie des Business Models du logiciel libre

I. Introduction

Dans cette troisième partie nous allons traiter du cœur de la problématique de ce mémoire. Les deux premières parties ont été très utiles pour situer le cadre de la problématique des Business Models. En effet ; elles nous ont permis de mieux définir le logiciel libre, en particulier ses aspects techniques, juridiques et institutionnels ainsi que de comprendre ses fondements économiques, en particulier son mode de production coopératif, source de ses qualités.

Dans cette partie nous allons donc proposer une typologie des Business Model des entreprises qui gravitent autour du phénomène du logiciel libre.

Dans un premier temps nous allons définir la notion de Business Model, puis nous verrons les points communs sur lesquels reposent tous les Business Model du Libre.

Dans un deuxième temps, nous présenterons les cinq grands types de Business Model que l'on peut trouver dans le Libre. Pour chaque Business Model nous ferons d'abord une présentation de celui-ci, puis nous illustrerons nos propos par un exemple détaillé d'utilisation de ce modèle. Pour réaliser cette étude nous nous appuierons sur des sources diverses : le Web et en particulier les sites Web de projet de logiciel libre, des sites de références dans le communauté du Libre (www.Slashdot.com ou www.linuxfr.org), des échanges avec des acteurs du Libre (développeurs, cadres de SSII) que ce soit en face à face ou par Internet, et enfin, la presse, grâce à la lecture de la presse spécialisés en informatique ou sur le Libre (Login, Linux Loader, Zdnet ou le journal du net).

Dans une dernière partie nous verrons que les différents Business Model du Libre sont interdépendants, et forment une dynamique que l'on peut considérer comme le Business Model global du Libre.

II. Présentation général

1. La notion de Business Model

Avant d'entamer une réflexion approfondie sur les Business Model du logiciel libre il est important de bien comprendre ce que l'on entend par la notion de Business Model.

Maitre et Aladji (1999) font remarquer que ce concept est particulièrement riche et fécond. Ils ajoutent que du fait de sa polysémie, la notion de Business Model est assez difficile à traduire. Ils font observer que la traduction par « modèle d'affaire » est incompréhensible. D'autres tentatives de traduction envisageable sont aussi à éviter : « modèle de développement » serait plus précis mais semble tout droit sorti d'un manuel d'économie du tiers-monde, « modèle stratégique » dégage une assez forte odeur de société de conseil et semble renier la dimension tactique inhérente à la temporalité du modèle. Maitre et Aladji (1999) en concluent que faute de mieux, qu'ils s'en tiendront volontiers à « modèle économique » ou plus précisément pour les start-up à « modèle de croissance », Dans le cadre de notre étude sur le logiciel libre, nous préférons nous aussi la traduction de « modèle de croissance »

Ce choix de la notion de « modèle de croissance » pour traduire Business Model est conforté par l'examen de la littérature managériale américaine à ce sujet. De cet examen Maitre et Aladji (1999) remarquent que l'on trouve la notion de business Model employée dans 3 sens différents :

- « *Le premier sens et le plus fréquent est l'acception simplifiée de ce que nous venons de décrire: le business model d'une entreprise est alors pour l'essentiel la structure de son offre, sa manière de générer des revenus, son organisation et la structure de coûts qui en résulte, sa manière de nouer des alliances adéquates et la position dans la chaîne de la valeur qui en résulte. Pour ce premier sens, l'expression « modèle de croissance » est tout à fait adaptée.*

- *Un usage actuellement très à la mode diffère de façon significative de ce sens initial: il concerne les sociétés opérant, d'une manière ou d'une autre, dans le secteur de l'Internet. Pour celles-ci en effet, le business model désigne le plus souvent leur seule manière de générer des revenus. Il est vrai que sur ce diable de réseau, le mode de génération des revenus n'est pas toujours un problème trivial.*

. *Enfin rappelons l'acception donnée à ce terme par les analystes financiers: lorsqu'ils rendent compte de la structure du compte de résultat d'une entreprise, ils appellent business model la décomposition du compte de résultat en pourcentage des ventes. »*

Dans la suite de ce mémoire quand nous allons parler de Business Model, c'est bien dans le sens de « modèle de croissance ». Il s'agira bien de voir qu'elles sont les principes qui permettent aux entreprises du Libre de se développer et surtout d'assurer leur rentabilité à moyen ou long terme.

2. Les points communs des Business Models du Libre

Un Business Model pour être viable implique, qu'au moins à moyen terme, les coûts soient inférieurs aux recettes. Pour arriver à cet équilibre il existe deux stratégies viables, soit de concentrer son effort dans la réduction des coûts soit de le concentrer dans l'augmentation des recettes. L'industrie du logiciel propriétaire suit plutôt cette deuxième logique alors que le monde du libre suit plutôt la première.

En effet, le principe des licences d'utilisation permet dans l'industrie du logiciel propriétaire aux éditeurs d'avoir des recettes importantes. Mais si les recettes sont importantes les coûts sont aussi très élevés du fait du mode de production hiérarchisé et rémunéré ainsi que des dépenses de marketing. En effet, les caractéristiques économiques du logiciel en particulier les effets de clubs, incitent les firmes éditrices à dépenser énormément en marketing pour imposer rapidement leurs produits, pour bénéficier ensuite d'une rente de monopole.

Au contraire un des traits communs des Business Models du logiciel libre est qu'ils sont basés sur de faibles dépenses. En effet, comme nous l'avons vu dans la

deuxième partie, le modèle du Bazar permet de produire des logiciels de très hautes qualités avec des coûts très bas, voire nuls. Les seuls coûts sont les coûts de coordination pour les projets de très grandes tailles : la gestion d'un projet qui implique des milliers de développeurs induit un minimum de coûts. Mais ces coûts sont très faibles puisque les développeurs ne sont généralement pas rétribués pour leurs contributions. Les mécanismes d'incitations ne sont pas basés sur l'argent, mais principalement sur la reconnaissance par la communauté.

De plus, il n'y a pas ou très peu de dépenses en marketing dans le Libre. Le marketing du Libre est un marketing viral, c'est-à-dire qu'il fonctionne sur le principe du bouche à oreille et de la réputation. L'outil Internet qui joue un rôle vital dans la coordination des développeurs, joue ici aussi un rôle fondamental. En effet, c'est bien au travers de l'Internet que se propage ce marketing viral qui permet aux meilleurs projets de se faire connaître extrêmement rapidement. Cette forme de marketing outre qu'elle soit gratuite, est particulièrement efficace, car très rapide et surtout très ciblée. Cependant, elle est à double tranchant : les fausses rumeurs peuvent très rapidement briser l'image d'un projet.

Il existe de nombreux modèles de business dans l'industrie du Libre, comme nous allons le voir dans la suite de ce mémoire. Mais ils ont tous un point commun : ils reposent toujours sur une bonne maîtrise des coûts. Il est plus facile d'être rentable quand on dépense peu. Surtout, cela permet d'imaginer des Business Model qui n'auraient pas été envisageables avec des coûts importants.

III. Typologie des Business Models du Libre

1. Le modèle non business

a. Présentation du modèle

Le modèle non business est certes un modèle de croissance mais il n'est pas appliqué à une entreprise, c'est en cela qu'il n'est pas business.

Il peut paraître provoquant de commencer une typologie des Business Model par les modèles non business. Cependant, ce choix est justifié par le fait que c'est de loin le modèle le plus répandu et le plus emblématique du logiciel libre. De plus, même si ce modèle est « non business », il joue un rôle économique important.

La plupart des logiciels libres réputés sont battis autour de ce modèle de croissance. Ce modèle est celui de logiciels aussi réputés que le noyau de Linux, qu'Apache, ou des langages de programmation comme PHP.

Le développement de ces logiciels n'est pas assuré par une entreprise mais par une association ou par une fondation à but non lucratif. C'est le modèle que décrit Raymond (1998) quand il évoque le Bazar. Une personne ou un groupe décide de produire un logiciel. Ils laissent le code ouvert, et chacun peut l'améliorer à condition de respecter les licences du logiciel libre qui lui sont appliquées. C'est le mécanisme que nous avons décrit dans les deux premières parties de ce mémoire.

Une fois le projet assez avancé, une fondation à but non lucrative est chargée d'assembler, de coordonner et de mettre ensemble les travaux des développeurs. Donc ce modèle de croissance est basé sur une **communauté** de développeurs qui participent à un **projet**, sous la coordination d'une **fondation**.

Mais si ce modèle de croissance ne peut pas être vraiment qualifié de Business Model, il a une place légitime dans une typologie des Business Model du Libre. D'abord du fait de sa prépondérance, et du fait qu'il produit des logiciels qui sont sur le marché et qui concurrencent de façon sérieuse des logiciels source du Business Model d'entreprises parfois très importantes. Ainsi, Linux concurrence directement les systèmes d'exploitations Windows de Microsoft, le langage PHP concurrence le langage propriétaire ASP.... Ces projets sans Business Model, mais avec un vrai modèle de croissance jouent un rôle économique important.

De plus, si les fondations qui coordonnent les projets de logiciels libres sont à but non lucratif, elles ont cependant besoin d'un minimum d'argent pour fonctionner. Cet argent vient principalement sous forme de dons d'autres entreprises qui vivent du logiciel libre en particulier des entreprises de services ou de hardware. Ainsi des entreprises comme IBM, Sun Microsystem, HP, qui aujourd'hui vendent des solutions hardwares livrés avec des logiciels libres préinstallés, subventionnent certains projets de logiciel libre.

En outre, ce modèle de croissance ne peut exister que s'il est inscrit dans un système économique qui lui est favorable. Ce modèle de croissance est intimement lié à au Business modèle des services et du hardware que l'on traitera plus loin. En effet, les développeurs qui travaillent sur des projets libres ne sont pas rémunérés pour cela. Beaucoup d'entre eux réalisent ce travail pendant leurs loisirs, mais d'autres au contraire sont employés par des sociétés qui proposent des services dans le domaine du logiciel libre.

Le modèle non business, s'il présente un modèle de croissance efficace, ne peut exister seul et doit s'intégrer dans un système, qui permet de proposer des méthodes d'incitations pour des développeurs bénévoles.

Nous allons étudier plus en détails le modèle de croissance d'Apache, pour illustrer le fonctionnement du *model non business*.

b. Apache

- **Présentation**

Apache est un serveur HTTP maintenu par l'*Apache Foundation*. Un serveur http est un logiciel qui permet d'utiliser un ordinateur comme serveur Internet.

« Apache est le fruit de l'effort de webmasters qui développèrent chacun des extensions au serveur le plus populaire de l'époque (le serveur HTTP du NCSA), puis décidèrent de former l'Apache Group afin de mettre en commun leurs efforts et de produire le serveur HTTP le plus robuste et le plus stable jamais connu. » source April ¹¹

Il existe une mythologie naissante sur l'origine du nom Apache. Certains disent que cela vient de « a patchy server » à cause des nombreux patches (correctif) du début. D'autres d'une manière plus sérieuse que les instigateurs de ce projet ont pris ce nom en mémoire des Apaches pour leur grande adaptabilité sur le terrain.

Le projet Apache est géré par une Foundation à but non lucratif la *Apache Software Foundation*¹²

¹¹ <http://www.april.org/groupe/labo/apache>

¹² <http://www.apache.org>

- **Bref historique**

Le développement d'Apache a commencé en 1994. Brian Behlendorf, alors âgé de 21 ans, était responsable d'un des principaux serveurs Internet commerciaux d'Internet aux Etats-Unis : le serveur du site Web du magazine HotWired.

Ce serveur, comme la plupart, utilisait alors le logiciel basé sur Unix écrit au centre national des applications des superordinateurs (NCSA) à l'université de l'Illinois. (Le seul produit concurrent était alors le serveur développé au CERN.)

Le NCSA a distribué le code source gratuitement et possédait un groupe de développeurs impliqués activement à améliorer le code source en liaison avec les premiers utilisateurs. Alors que Behlendorf et d'autres développeurs programmaient des corrections ou des « patches » pour le serveur du NCSA, ils partageaient aussi à travers les mailings-lists leurs points de vue sur l'évolution de l'Internet.

Cependant Behlendorf et beaucoup d'autres utilisateurs, ressentait des frustrations grandissantes devant les réticences de l'équipe du NCSA de prendre en compte leurs suggestions. En fait, à cette période, un certain nombre des membres de l'équipe du NCSA avait quitté le projet pour fonder Netscape, de plus l'université était entrain de négocier avec des entreprises commerciales pour leur abandonner la licence de leur logiciel serveur. En réaction, Behlendorf et six autres développeurs décidèrent de créer une mailing-list pour collecter et intégrer les patches au logiciel serveur du NCSA. Ils décidèrent d'un commun accord qu'il s'agirait d'un processus collégial. Donc si de nombreuses personnes pouvaient suggérer des modifications, seulement un petit nombre pourrait réellement modifier le code. En août 1995, le groupe présentait la version 0.8 d'Apache, qui était substantiellement différente du programme initial et qui constituait une base de travail solide pour commencer le projet. Une différence notable était l'API (Application Programm Interface) qui permet au développement d'Apache d'être très modulaire. Cette avancée conceptuelle permit la modification de certaines parties du programme sans que cela n'ait de répercussions sur l'ensemble. C'est un élément très important dans le cadre d'un développement du type Bazar.

- **La communauté Apache**

Le « coeur » Apache autour des quelques fondateurs s'est accru petit à petit par

cooptation. Les règles initiales ont été édictées par le groupe fondateur et évoluent au fil des votes. La communauté Apache est une communauté mature comme la définit Rastetter (2002), c'est-à-dire une communauté où l'on peut remarquer qu'il y existe un noyau de partenaires multiples et équilibrés.

La gestion de source est faite par un répertoire (*repository*) de version courante (CVS), outil de développement coopératif classique. Tout le monde a accès en lecture aux clichés (*snapshot*) sur la base commune de gestion de la version courante.

Les changements de code sont proposés sur les listes et les membres actifs votent. Les contributeurs échangent une moyenne de 40 messages par jour sur tout type de sujets: les bogues, les nouvelles fonctionnalités, les problèmes des utilisateurs. . .

Le développement du code se fait sur les machines personnelles des développeurs.

Les propositions de mise à jour sont effectuées selon un format standardisé et inscrites au répertoire CVS. Chacun peut voter, mais seules comptent vraiment les contributions de ceux qui sont reconnus comme étant des experts sur le sujet traité. Les véto doivent être solidement argumentés.

La cooptation se fait quand une personne active est distinguée par un membre, en pratique après une participation active de six mois.

- **La « Success story » d'Apache**

L'Apache Software Foundation est à but non lucratif, donc elle n'a pas de Business Model ayant pour but d'engranger des bénéfices. Les dons, qu'elle reçoit que ce soit de particuliers ou d'entreprises, sont suffisants pour assurer son bon fonctionnement et en particulier son rôle coordinateur. Son modèle de croissance est efficace puisqu'il lui permet d'assurer son développement.

De plus, Apache est très souvent donné comme exemple de logiciel libre qui a su s'imposer face à des concurrents propriétaires. En effet, en Août 2002, Apache détenait 64% du marché des serveurs HTTP, contre moins de 25% pour son seul réel concurrent

Microsoft et moins de 5% pour les autres acteurs du marché comme iPlanet ou NCSA. (Source Nercraft¹³)

¹³ <http://www.netcraft.com/survey/>

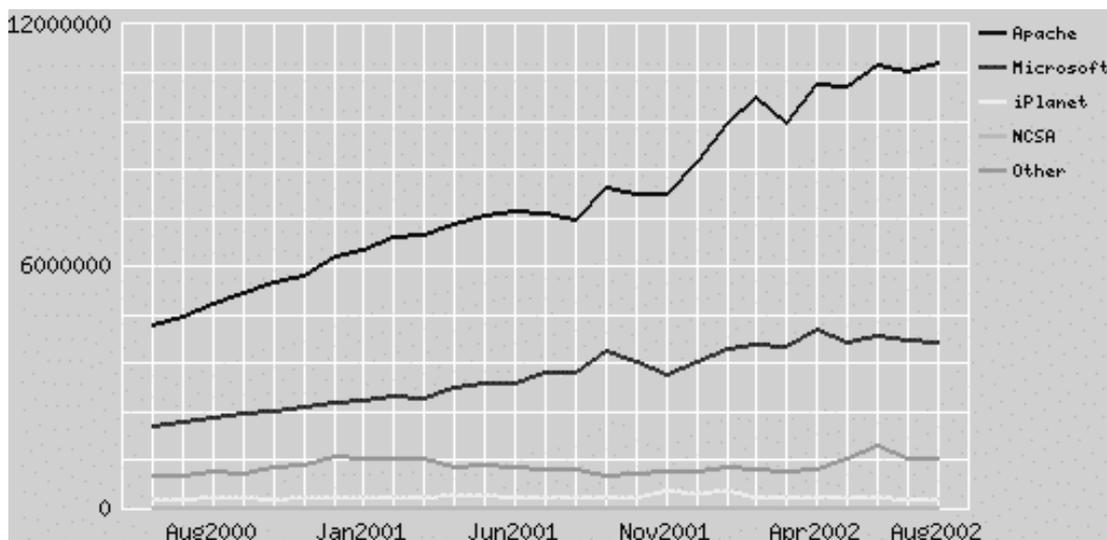


Fig 5. Parts de marchés des Serveurs HTTP
Source Netcraft¹⁴

- **Conclusion**

Le projet Apache est un exemple de la viabilité du modèle de croissance « modèle non business », c'est un projet qui continue d'évoluer, la fondation ne perd pas d'argent et le logiciel est leader sur son marché. Ce modèle de croissance qui est trop souvent identifié au Business Model du Libre en général, n'est pas le seul qui existe dans le logiciel libre. En outre, sa viabilité n'est possible uniquement s'il existe des entreprises qui acceptent de le soutenir. Nous allons désormais nous intéresser à ces entreprises qui n'éditent pas de logiciel, mais sans qui le logiciel libre n'existerait pas.

2. Le modèle du service et du hardware

a. Présentation du modèle

Il s'agit du Business Model des entreprises qui proposent des services et du conseil en informatique la seule particularité c'est que ces services portent sur du logiciel libre. De nombreuses petites entreprises d'ingénierie informatique existent sur ce créneau. De même, le business modèle « hardware » est celui des firmes qui vendent du matériel informatique avec des solutions libres préinstallés. Des entreprises comme IBM

¹⁴ <http://www.netcraft.com/survey/>

ou plus récemment Sun propose des solutions informatiques basées sur des logiciels libres, en particulier Linux ou Apache.

Les Business Models de ces types n'ont rien d'originaux et sont tout à fait comparables à celles de leurs concurrentes qui travaillent avec des logiciels propriétaires. Il existe pourtant une différence importante ces entreprises se doivent de participer à la communauté du Libre. Si les entreprises commencent à avoir des comportements opportunistes et cherchent à utiliser le mécanisme du Libre à leur seul profit, les développeurs bénévoles pourraient se sentir lésés et dupés et ne participeraient plus aux différents projets en Open Source. Foray et Zimmermann (2002) montrent d'ailleurs que la plus grande menace pour le mouvement du Libre est la démotivation des développeurs si leur travail est récupéré par des entreprises à leur seul profit. Ils montrent en particulier que ce sont les développeurs les plus efficaces, qui dans un tel cas de figure, quitteraient les premiers le mouvement du Libre car ce sont eux qui ont le plus à perdre.

Les entreprises de service et de hardware qui ont basé tout ou partie de leur Business Model sur l'essor du logiciel libre, ont bien compris cet écueil et cherchent par tous les moyens à montrer leur bonne volonté à la communauté des développeurs de logiciels Open Source. La plupart d'entre elles apportent leur soutien financier à des fondations qui coordonnent des projets de logiciels libres. Les plus grandes structures emploient des développeurs pour qu'ils travaillent sur des projets Libres. Nous avons vu que la des grandes forces du logiciel libre c'est son mode de production coopératif. En embauchant des développeurs pour qu'ils travaillent sur des projets de logiciels libres, elles permettent la pérennisation de ce système. D'abord directement du fait de la contribution qu'apportent ses développeurs, et indirectement en proposant des perspectives d'embauches aux développeurs bénévoles (« *career concern incentive* »).

Pour l'entreprise, cette mise à disposition d'une partie de sa main d'œuvre, a un triple avantage.

- Elle signale sa bonne volonté à la communauté du Libre
- Elle permet d'assurer la pérennité du logiciel libre qui est à la source des ses revenus.
- En participant à ces projets, l'entreprise acquiert des compétences qui lui seront particulièrement utiles dans ses missions de services.

- Cette participation a aussi un intérêt marketing. Dans le monde du Bazar, la réputation joue un très grand rôle, et le prestige d'un développeur particulièrement brillant rejaillira sur son employeur, et pourra servir d'argument marketing auprès des clients et des prospects. Il fonctionne en fait sur le principe du marketing de parrainage.

La viabilité du Business Model du service et du hardware, n'est plus à montrer. Depuis de nombreuses années il existe avec succès dans le monde des logiciels propriétaires. Il fonctionne sur le même principe dans le cadre du Logiciel Libre si ce n'est la nécessaire contribution des entreprises au mouvement du libre. Si cette contribution signifie certes des coûts supplémentaires, elles apportent aussi des avantages non négligeables à l'entreprise en terme de marketing, et d'expertises. Cette contribution de la firme au Libre ne suffit pas à modifier fondamentalement le Business Model classique du service et du hardware, qui lui, a montré sa viabilité. Par contre, ces firmes devront veiller à ne pas trop dépenser dans leur soutien au Libre, pour ne pas se mettre en danger. Il est important de trouver le bon équilibre entre implication dans le mouvement Open Source et l'obligation d'être rentable.

b. Alcove

Alcove est une société française d'expertise et de services en informatique aux entreprises. Elle est spécialisée dans des solutions à base de logiciel libre. Elle se définit elle-même comme étant « *La première société européenne en informatique Libre* ». La société a été créée en 1996 par Christophe Le Bars et Lucien Petit et possède une filiale en Allemagne.

La clientèle d'Alcove se compose principalement de Grands Comptes et d'acteurs du marché informatique (Constructeur, éditeur...). L'intérêt des Grands Comptes pour le logiciel libre s'explique par leur volonté de reprendre en main leur système d'information.

Alcove fournit pour les logiciels libres des services qui sont traditionnellement ceux des éditeurs : formations de haut niveau, customisation du produit, maintenance...

Alcove est une entreprise de service informatique qui a les mêmes caractéristiques que toutes les entreprises en expertises en informatique. Elle connaît certes, des

difficultés financières puisque l'entreprise est en redressement judiciaire. Cependant, les dirigeants précisent que cette mauvaise situation est due à des causes conjoncturelles et non structurelles. La situation actuelle serait la conséquence d'investissements trop importants et au retournement du marché des nouvelles technologies à la fin 2001. Le redressement judiciaire est le résultat d'un endettement trop important. Cependant, cette mauvaise passe ne remet pas en cause ni la viabilité de la société Alcove ni celle de son Business Model. En effet, le chiffre d'affaire de la société est en constante augmentation, et sans le poids de la dette, l'activité d'Alcove est bénéficiaire.¹⁵

Donc si la société Alcove connaît des problèmes passagers, la viabilité de son Business Model n'est pas en cause. Si Alcove a un Business Model similaire de celui de ces concurrentes du monde propriétaire, comme ces dernières, elle est touchée par la crise qui frappe l'industrie informatique ; Alcove diffère des entreprises de services informatique par son engagement dans le Libre.

Pour Christophe Le Bars co-fondateur d'Alcove et directeur technique, un des objectifs prioritaires de sa société est « *rester en symbiose avec la communauté du Libre* ». D'ailleurs, Alcove apporte un soutien effectif à la communauté du Libre puisque 20% du temps de travail global de l'entreprise est consacré à des développements pour des projets de logiciel libre. Alcove a d'ailleurs créé un site différent de son site officiel pour présenter les différentes contributions de ses développeurs aux logiciels libres. (<http://www.alcove-labs.org>). On s'aperçoit que les développeurs d'Alcove investissent beaucoup de leur travail dans le développement de la distribution linux indépendante « Debian ».

On peut constater que le Business Model d'Alcove reprend bien les caractéristiques du Business Model « service et hardware » que nous avons décrit précédemment : Business Model classique des entreprises de services informatique accompagné d'un investissement important dans le soutien de la communauté du libre. Même, si la société Alcove connaît quelques difficultés financières, elles semblent plus être le fruit d'erreurs de gestion et du retournement de la conjoncture économique que du fait d'un Business Model défaillant.

¹⁵ http://www.linuxfrench.net/article.php3?id_article=948

3. Le modèle des distributions Linux

a. Définition d'une distribution Linux

Avant de nous pencher sur la le Business Model des distributions Linux, nous allons présenter rapidement ce qu'est une distribution Linux. Pour, bien comprendre ce qu'est une distribution, il est important de voir comment fonctionne Linux.

- **Présentation de Linux**

Quand on parle du Logiciel libre, tout le monde pense à Linux. Le programme de Linus Torwalds est devenu le fer de lance du mouvement du Libre.

La définition la plus courte pour décrire Linux pourrait être *un système d'exploitation de la famille des Unix*.

- *Un système d'exploitation*

Un ordinateur, en soi, n'est rien de plus qu'un ensemble de composants. Le système d'exploitation est un groupe de programmes qui donnent vie à votre machine. Un système d'exploitation comprend des programmes de différents niveaux. Certains gèrent le matériel (lecteur de disquettes, clavier, souris, mémoire, etc.). Ce matériel est géré par des programmes bas niveau (bas en ce sens qu'ils sont plus proches du matériel) contenus, entre autres, dans le noyau. A cela s'ajoutent de nombreux utilitaires permettant, par exemple, la copie de fichiers ou l'édition de textes. Enfin, à un niveau encore plus haut se trouvent des programmes permettant par exemple de faire de la bureautique ou de naviguer sur l'Internet.

Il faut retenir ici qu'un système d'exploitation est composé de programmes gérant le matériel et d'utilitaires, divers et variés, et que ce tout est une base nécessaire au fonctionnement d'un ordinateur. Les systèmes d'exploitations le plus répandus sont les Windows de Microsoft.

- *Un Unix*

Unix est un système d'exploitation conçu en 1969 par deux chercheurs (Thompson et Ritchie) des laboratoires Bell. À ses débuts, Unix est un produit de laboratoire, distribué gratuitement aux universités. Il remporte un grand succès. En 1975, l'engouement est tel que les laboratoires Bell décident de le

commercialiser. Disponible sous forme de licence (on achète une licence, qui permet de le modifier et de le revendre), de nombreuses déclinaisons apparaissent. Mais cette diversité de systèmes Unix crée des problèmes de compatibilité. Les principaux acteurs décident alors de créer différentes normes (POSIX, UNIX95) assurant la compatibilité des différents systèmes Unix.

En 1991, Linus Torvalds, alors étudiant de l'université d'Helsinki, se lance dans un projet fou : écrire un système d'exploitation libre pour PC compatible avec les systèmes Unix : **Linux**, *l'Unix de Linus est né*.

Le projet Gnu/Linux qui est à la base du système d'exploitation Linux concerne essentiellement le noyau de Linux. Linux comprend un noyau (Kernel) qui est le centre nerveux du système. Il permet de faire le lien entre les programmes et le matériel et en fait un système multitâche grâce au partage des ressources. Chaque noyau possède une version composée de 3 numéros séparés par un point (par exemple 2.2.10). Le deuxième numéro permet de savoir si la version est stable (numéro pair) ou en cours de développement (numéro impair). Le troisième permet de savoir si la version est récente (plus le numéro est grand, plus la version est récente).

- **Les distributions Linux**

Pour faire de Linux un système d'exploitation convivial il faut au moins lui adjoindre une interface graphique. Généralement, les systèmes d'exploitation ne se contentent pas d'être des logiciels qui assurent l'interface entre le hardware et les autres logiciels. Ils comprennent aussi de nombreux petits outils logiciels : des applications graphiques simples, des logiciels traitements de textes rudimentaires, etc...

Une distribution Linux c'est donc un système d'exploitation complet. Il existe de très nombreuses distributions de Linux. Le magazine en ligne « linux.org » qui essaie de recenser toutes les distributions n'en dénombre pas moins de 165. Ces distributions sont plus ou moins complètes certaines ne proposent même pas d'interface graphique, d'autres sont extrêmement complètes. Ces dernières sont bien plus qu'un simple système d'exploitation, puisqu'elle comprennent de véritables logiciels applicatifs qui recouvrent tous les besoins d'un utilisateur ordinaire.

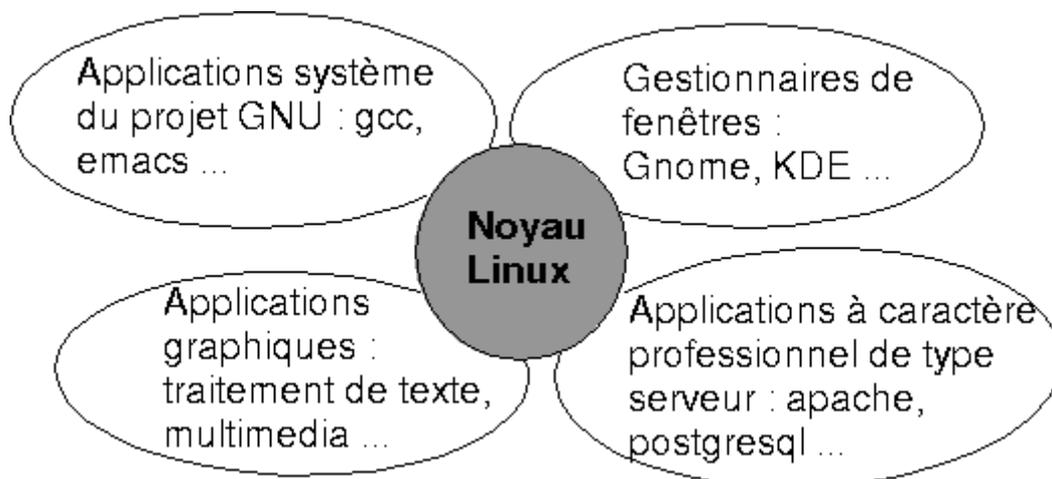


Fig 6. Les principaux composants d'une distribution Linux
Source Rapport sur GNU/Linux, LE DISEZ Erwan¹⁶

Ces grandes distributions (Red Hat Suze Mandrake...) intègrent dans leur Linux de nombreux logiciels libres et parfois même des logiciels propriétaires.

b. Présentation du Business Model

Les principales distributions Linux (Mandrake, RedHat...) sont aussi des entreprises qui peuvent même être cotées en bourse. Nous allons voir le Business Model de ces entreprises qui repose sur des distributions Linux.

La structure administrative de l'entreprise ne se confond pas avec celle du projet de logiciel libre qu'est la distribution. La distribution est développée comme tout logiciel libre sur le modèle du Bazar, la seule différence est que la société coordonne le projet. La société emploie aussi de nombreux développeurs qui participent à ce projet. Cependant la majorité des développeurs qui travaillent sur une distribution ne sont pas employés par l'entreprise. Pour la société à l'origine d'une distribution Linux, si les coûts sont inférieurs à ceux d'une société éditrice de logiciels propriétaires, ils sont relativement conséquents. Pour que leur Business Model soit viable il leur est donc nécessaire d'avoir des recettes importantes.

Ces recettes peuvent être de différentes natures.

¹⁶ http://erwan.ledisez.free.fr/docs/rapport_freesoftware/index.html

- *Les services* : Les sociétés «éditrices» de distributions Linux généralement proposent des services (formations, installations, maintenance...) liés à leurs distributions.
- *Version OEM* : une autre source de revenue est l'association à un constructeur de hardware, qui vend son matériel informatique avec une version de la distribution Linux préinstallée. Le constructeur verse une certaine somme à la société éditrice de la distribution linux.
- *Les boites* : Les sociétés peuvent mettre en vente leurs distributions sous forme de « boites ». Le contenu de ces boites peut être très variable. La boite peut contenir simplement les CR-Roms de la distribution ou encore un livre d'aide ou un abonnement d'un an gratuit à la hotline de la distribution. Le contenu des CD-Roms peut aussi être très variable. En effet, les CD-Roms peuvent contenir une version basique de la distribution, ou une distribution qui intègre de très nombreux logiciels libres et parfois même propriétaires.
- *Les recettes annexes* : Outre la vente des « boites » et des services, les sociétés de distribution de Linux ont divers systèmes qui leur permettent d'augmenter leur revenu. Il peut s'agir de la vente de produits à l'effigie de leur distribution (T-shirt, casquette,...) ou alors l'appel à des dons. Un autre moyen de gagner de l'argent pour ces sociétés est de créer un « club » des utilisateurs dont la participation est payante, mais qui donne droit à des avantages (téléchargement de logiciels, informations sur les sorties des dernières versions des distributions...

Actuellement ces sociétés tirent leur revenus principalement de la vente de distributions en boite. Ce Business Model basé sur la vente de distributions en « boite » est particulièrement fragile et peu innovant. En effet, il est calqué directement sur le Business Model du logiciel propriétaire, où les entreprises éditrices vendent leurs logiciels en CD-Roms dans des boites. Cependant, les distributions Linux sont des logiciels libres. En conséquence, il n'y a pas de système de licence d'utilisation, donc il est tout à fait légal pour quiconque de faire des copies et des les distribuer gratuitement ou même contre rétribution. Le danger est qu'une distribution concurrente se crée à partir

de votre propre distribution. Ainsi, la distribution Mandrake a été créée directement à partir de la distribution RedHat.

Les distributions en boîte contiennent de plus en plus souvent, des services gratuits comme l'accès gratuit à la Hotline de la distribution pendant un an. Parfois les distributions comportent des logiciels propriétaires et de ce fait il n'est plus possible de les distribuer librement. Cependant, ces ventes de « boîte » ne semble pas être la base de la création d'un Business Model viable, d'ailleurs toutes les distributions annoncent qu'elles vont recentrer leur activité pour se concentrer sur leurs offres de services ou leurs alliances avec des sociétés de hardware, qui semblent être la source d'un Business Model plus intéressant.

Le Business Model de la distribution Linux basé sur la vente de « boîte » dans une logique, proche de celle du logiciel propriétaire, mais originale dans le monde du Libre, semble être abandonné par les sociétés de distribution Linux au profit du modèle dominant du Libre : le « Modèle service et hardware », que nous avons présenté précédemment.

Nous assistons donc vers une évolution du Business Model des distributions Linux qui se confond de plus en plus avec celui du « modèle service et hardware ». La différence notable, est que dans le cas du modèle de la distribution l'offre de service et l'effort en terme de contribution active à la communauté du Libre se concentre sur la distribution. En effet, la société va proposer des services informatiques en support et en formation, principalement sur sa propre , distribution linux et son investissement dans la communauté du Libre va se concentrer encore une fois sur sa distribution.

c. Mandrake

- **Linux-Mandrake**

Le système d'exploitation Linux-Mandrake a été créé en 1998 pour rendre plus simple l'utilisation de Linux. La Linux-Mandrake a été conçu à partir de la distribution Red Hat en y ajoutant l'interface de fenêtrage KDE, ce qui a permis, dès son origine, à Linux-Mandrake d'être reconnue comme étant une des distributions les plus simple à utiliser pour un néophyte de Linux.

A un système d'exploitation puissant et stable, mais qui demandait de la part de l'utilisateur de bonnes connaissances en informatique et des rudiments de programmation via une interface en mode texte appelée "ligne de commande", MandrakeSoft a associé un environnement de travail graphique.

D'ailleurs, le point fort de Linux-Mandrake reste sa simplicité d'installation d'utilisation et de paramétrage.

L'installation automatisée de Linux-Mandrake est reconnue comme étant la meilleure procédure d'installation de Linux. Linux-Mandrake est le système d'exploitation Linux qui offre la détection matérielle la plus avancée et la plus efficace. Cette excellence dans la détection facilite, grandement l'installation du système et permet gagner de beaucoup de temps. En outre, Linux-Mandrake est le système d'exploitation Linux qui pousse le plus loin l'intégration et l'utilisation d'une interface utilisateur fortement graphique, faisant ainsi mentir l'image d'un système Linux géré uniquement en ligne de commande ou en mode texte.

Par nature, il est très difficile d'évaluer l'importance exacte de la diffusion d'une distribution Linux. La distribution Linux-Mandrake est une des distributions les plus répandues avec la Red-Hat, Suze, ou Debian.

- **MandrakeSoft**

MandrakeSoft est la société qui édite le système d'exploitation Linux-Mandrake sur le mode du développement ouvert (accès libre au code source publié selon les termes de la GPL). Depuis le 30 juillet 2001, la société est cotée en bourse, au Marché Libre d'Euronext Paris. MandrakeSoft emploie environ une centaine de personnes.

MandrakeSoft propose deux gammes de produits : la gamme entreprise et la gamme grand public.

Pour le grand public MandrakeSoft propose diverses versions de Linux-Mandrake en « boîte », ces versions intègrent différents logiciels. Elle propose aussi des services pour les particuliers (support technique, hotline...). Pour le Marché des entreprises, MandrakeSoft vend d'autres versions de sa distribution Linux pour les besoins spécifiques des entreprises (Version Serveur, version Firewall...). Elle offre aussi une gamme complète de services aux entreprises (Support technique, formation, Consulting, Hotline...)

A travers son site Web MandrakeStore¹⁷, elle vend aussi des produits associés comme des T-shirts, des casquettes à l'effigie du logo de la distribution.

Enfin elle, a créé le MandrakeClub¹⁸, l'entrée dans ce club est payante et constitue un moyen pour les soutiens de Mandrake de faire des dons financiers. En outre, ce club, donne aussi accès à certains avantages pour ses membres comme la possibilité de télécharger des logiciels complémentaires pour leur distribution Linux.

Actuellement MandrakeSoft réalise près de 85% de son chiffre d'affaire avec la vente de distribution en « boîte ». Cependant, MandrakeSoft veut diminuer sensiblement l'importance de la part de des ventes des « boîtes » dans son chiffre d'affaire. *«Aujourd'hui, MandrakeSoft réalise 85% de son chiffre d'affaires sur la vente de boîtes, son objectif est désormais d'orienter son dynamique de développement vers les services, qui devraient représenter 40% de son chiffre d'affaires en 2002 et 50% en 2003»¹⁹* Les boîtes devraient garder surtout un intérêt marketing, pour assurer la visibilité de Linux-Mandrake mais perdre leur rôle prépondérant dans le chiffre d'affaire.

Les dirigeants de MandrakeSoft, après avoir axé leurs efforts sur le marché grand public se retournent vers le marché « entreprise », car ils y voient de meilleures perspectives de croissances. D'ailleurs, au cours du premier semestre 2002 l'entreprise a connu des difficultés financières, du fait du retournement de la conjoncture en 2001, d'erreurs stratégiques et enfin, des faiblesses du Business Model de la vente de « boîte ».

MandrakeSoft se repositionne désormais plus vers son marché entreprise en proposant d'avantages de services aux entreprises mais aussi en cherchant à signer des contrats OEM, c'est-à-dire de signer des partenariats avec des constructeurs informatiques pour qu'ils intègrent la distribution Linux-Mandrake dans leurs solutions hardware.

MandrakeSoft comme ses concurrents, est entrain d'abandonner le Business Model propre aux distributions Linux et basé sur la vente de « boîtes » pour adopter le Business Model « services et hardware ».

¹⁷ <http://www.mandrakestore.com>

¹⁸ <http://www.mandrakeclub.com>

¹⁹ Marie-France Souloumi responsable distribution chez Mandrake Soft

4. Le modèle défensif

a. Présentation du modèle

Le modèle défensif pourrait aussi se nommer le modèle de la dernière chance. En effet, il est adopté par des entreprises éditrices du logiciel libre qui sont en difficulté face à la concurrence et qui se retournent vers ce modèle de croissance en espérant rebondir.

Le principe de ce Business Model est simple. L'entreprise éditrice d'un logiciel propriétaire offre les codes sources de son logiciel à la communauté du Libre. Il se constitue un projet de logiciel libre autour de ses sources qui fonctionne de façon classique sur le modèle du Bazar de Raymond (1998). Ce logiciel évolue, il a les caractéristiques de tous les logiciels libres : faible coûts de productions, qualité, pérennité... A partir de là il existe deux logiciels très semblables : le logiciel Open Source et son cousin propriétaire. En effet, la société éditrice peut utiliser le logiciel Open Source pour son propre logiciel. La société éditrice apporte des compléments à la version propriétaire, et peut alors la vendre sous licence, ou adopter d'autres Business Model du monde propriétaire (Freeware).

Cependant, ce modèle de croissance a de nombreuses limites. D'abord, la transition entre un logiciel propriétaire et un logiciel libre peut s'avérer délicate. La communauté du Libre peut être réticente à participer à un projet qui reste lié à un logiciel propriétaire. En outre, il existe de nombreuses difficultés techniques pour transformer un logiciel propriétaire en logiciel Open Source. Il ne suffit pas d'ouvrir le code d'un logiciel propriétaire pour que la communauté du Libre puisse tout de suite s'en emparer pour le faire évoluer. En effet, les logiciels Open Source sont généralement construits sous forme de module ce qui permet un type d'organisation décentralisée du type du Bazar, alors que les logiciels produits par le principe de la Cathédrale, ne sont pas créés sous forme de modules.

Une autre difficulté de modèle c'est le contexte dans lequel il est appliqué. Il est généralement adopté par des entreprises qui ont de très faibles parts de marché. Du fait de l'effet de club qui caractérise l'économie du logiciel, il est très difficile de lutter contre un logiciel en position de quasi-monopole, et le passage à un système ouvert ne change pas grand-chose à la situation.

Pour l'entreprise éditrice, la viabilité de ce Business Model repose sur sa capacité de proposer la version propriétaire. Mais pour cela il faut que cette version propose des avantages vraiment significatifs par rapport à sa version libre. Cependant, c'est rarement le cas, et les utilisateurs se retournent vers la solution propriétaire.

On le voit le modèle défensif est un modèle difficilement viable, mais il est généralement adopté par des entreprises qui n'ont plus rien à perdre. Ce modèle avait été adopté par Corel pour sa suite bureautique Corel Office, qui était en concurrence avec celle de Microsoft, mais sans succès. La société Sun a aussi choisi ce modèle de croissance pour sa suite bureautique Star Office, qui reste propriétaire et commerciale et qui repose sur le logiciel libre Open Office. Si ce projet connaît un réel succès d'estime il n'est pas encore en mesure de lutter contre l'hégémonie de la suite Office de Microsoft. Un autre logiciel propriétaire a choisi ce Business Model pour rebondir, avec un bilan mitigé, c'est le navigateur Internet Netscape.

b. Netscape

Netscape a eu une place particulière dans l'histoire de l'industrie informatique. D'une situation de quasi-monopole sur le marché des navigateurs Internet, elle est passée à une situation d'acteur marginal. Ce changement radical de situation est dû à l'émergence sur ce marché d'un concurrent redoutable : Microsoft. Microsoft s'est servi de sa position hégémonique sur le marché des systèmes d'exploitation pour imposer son navigateur Internet (Internet Explorer) en l'intégrant dans son système d'exploitation Windows. Cette intégration lui a valu un procès devant les tribunaux américains, mais lui permet aujourd'hui de détenir plus de 95% du marché des navigateurs Internet. Pour lutter Netscape a choisi d'adopter le Business Model Open Source défensif.

En janvier 1998, l'éditeur Netscape offre à la communauté du libre les sources de son navigateur. Le projet de logiciel Open Source Mozilla est né.²⁰ Véritable première mondiale dans le monde de l'informatique, le don de Netscape constitue un tournant dans le mode de développement des logiciels, d'autant que cette action marque une réelle volonté d'inciter d'autres sociétés à investir dans l'Open Source. L'évolution du code de Navigator vers le libre apparaît comme logique. Cependant, il se compose aussi de parties appartenant à d'autres sociétés. Il faut alors les convaincre de le diffuser librement et en

²⁰ <http://www.mozilla.org>

cas de refus, amputer les programmes des lignes litigieuses Une nouvelle licence est mise au point, moins "libre" que la GPL la MPL Aujourd'hui, l'équipe de Mozilla la modifie afin d'augmenter sa compatibilité avec la GPL. Le travail de titan qu'impliquent le nettoyage et la réécriture du code (avec un nouveau moteur : Gecko) décourage beaucoup de développeurs. Il faudra trois années de travail à la communauté pour rendre Mozilla stable et utilisable. Aujourd'hui Netscape utilise le code de Mozilla pour son propre navigateur.

Mozilla ne se limite pas à des fonctions de simple navigateur. Il fait office d'éditeur de mails, de news, de pages Web avec un débogueur JavaScript incorporé D'autres modules existent tels que le "chat", la messagerie instantanée" le navigateur d'aide, le support des plug-ins traditionnels de Netscape (animations Flash, vidéos au format RealPlayer...) Mozilla supporte la quasi-totalité des standards du W3C, comme Math M. D'autres fonctionnalités comme le support LDAP dans le carnet d'adresses, prennent tout leur intérêt dans un cadre professionnel Chaque utilisateur est en mesure d'adapter Mozilla à ses besoins, grâce à sa flexibilité de configuration tant du côté de son interface graphique (thèmes, barres de menus personnalisables) que de la navigation (multiples possibilités de filtrages, gestion des certificats). Son installation graphique, très simple, fonctionne par module si vous le souhaitez, vous pouvez ne sélectionner que le navigateur. Le dynamisme impressionnant du projet lui a ainsi permis de s'imposer comme le navigateur libre le plus populaire.

Netscape en plus des fonctions de Mozilla propose des fonctions supplémentaires en particulier l'intégration de service proposé par sa maison mère AOL.

Cependant, l'adoption du modèle défensif n'a pas permis à Netscape de regagner des parts de marché bien au contraire. On peut expliquer ce manque de résultat par les problèmes de stabilité et le manque de fonctionnalités de la version 6 de Netscape. Ces problèmes étaient en grande partie dus aux difficultés rencontrées par le projet Mozilla de passer d'une architecture d'un logiciel propriétaire à celle d'un logiciel Open Source. Même AOL la maison mère de Netscape et premier fournisseur d'accès à Internet du monde a choisi Internet Explorer comme base de ses solutions de connexions à Internet. Cependant, nous pourrions être à la croisée des chemins pour Netscape. En effet, est sorti au mois d'août 2002 la version 7 de Netscape (en même temps que la version 1.0 (première version finalisée) de Mozilla). Cette version a été saluée par la presse

informatique, elle est très stable et propose plus de fonctionnalités que la dernière version d'Internet Explorer de Microsoft. AOL a décidé d'intégrer Netscape 7 dans son système de connexion Internet... Tous les éléments d'un nouveau départ sont réunis pour Netscape, s'il y parvient, il sera le premier à démontrer l'efficacité du modèle Open Source défensif, et il pourrait ouvrir la voie pour de nombreux autres logiciels....

5. Le modèle des logiciels « à façon »

a. Présentation du modèle

Le Business Model des logiciels à façon suit une logique un peu différente des autres Business Model du logiciel libre. En effet ce modèle ne repose pas sur le mécanisme du Bazar, comme la capacité de développer des produits de très haute qualité à faible coût, mais sur d'autres particularités du logiciel libre.

Un logiciel à façon est un logiciel qui est développé pour répondre à des besoins spécifiques d'une entreprise. Dans le monde de l'entreprise, les logiciels à façon jouent un rôle très important surtout dans les grands systèmes informatiques. Le fait que ces logiciels à façon soient développés en Open Source ne change pas la logique de son Business Model. En effet, comme il s'agit de logiciels très spécifiques, même s'il n'y a pas de licence, l'entreprise cliente, achète le logiciel. Elle n'achète pas une licence d'utilisation mais bien le logiciel en tant que tel. Ce genre de logiciels à façon est produit généralement par des développeurs salariés de l'entreprise éditrice de logiciel. En effet, les développeurs bénévoles ont peu d'intérêt à participer à ce genre de projet, car il s'agit de logiciels spécifiques, qu'ils ne peuvent pas l'utiliser.

Puisqu'il s'agit de logiciels spécialisés, l'impact de l'ouverture du code source est faible et le Business Model de l'entreprise éditrice est très proche d'un modèle propriétaire : développeurs salariés, vente du logiciel, et offre de services associés à ce logiciel. La viabilité d'un tel Business Model est connue dans le cas du logiciel propriétaire. Le fait de proposer des logiciels en Open Source apporte si peu de changements apparents : quels sont alors, les intérêts de l'approche « logiciel libre » dans le cas des logiciels à façon, par rapport à une approche classique basée sur des solutions propriétaires ?

Pour l'entreprise cliente les avantages sont nombreux. D'abord, l'ouverture des sources du logiciel lui assure en cas de faillite de l'entreprise éditrice de pouvoir se tourner vers une autre entreprise qui sera susceptible faire évoluer son programme puisque le code source est ouvert. En outre, toujours du fait de l'ouverture des sources, l'entreprise cliente, pourra apporter des modifications au logiciel en interne sans avoir à faire appel en permanence à la société éditrice. Pour l'entreprise cliente l'ouverture du code source lui permet d'assurer son indépendance par rapport à son fournisseur de solutions informatiques. De plus, un logiciel à façon libre n'est pas soumis à des licences d'utilisation, donc les coûts sont moins élevés que pour une solution propriétaire.

Pour la société éditrice il existe aussi des avantages. D'abord, le développement en Open Source lui apporte un avantage commercial non négligeable. En effet, elle peut faire valoir à ses clients potentiels que sa solution basée sur du logiciel libre apportera aux clients des avantages décisifs que nous venons juste d'énumérer. En outre, pour une entreprise éditrice, le logiciel à façon est un Business Model qui a prouvé son efficacité. En outre, nous avons vu que les mécanismes ne changeaient pas fondamentalement entre des logiciels libres et des logiciels propriétaires.

Cependant, il existe un risque très important pour la société éditrice de logiciels libres à façon. L'argument de l'indépendance de la société cliente est à double tranchant. En effet, comme le code source est ouvert, une fois le logiciel acheté, l'entreprise cliente peut continuer de le développer uniquement en interne ou faire appel à une autre entreprise. Alors que la partie service et support technique est aussi un moyen de compenser les coûts de développement du logiciel pour l'entreprise éditrice.

Un logiciel à façon est un logiciel qui sert à répondre à un besoin propre d'une entreprise. Cependant, il existe des points communs entre les différents besoins spécifiques des entreprises. Donc l'entreprise éditrice peut utiliser une base commune pour différents logiciels à façon et adapter cette base à chaque besoin spécifique de ses clients. Le risque alors est qu'une entreprise concurrente utilise votre code source pour produire un logiciel concurrent.

Ces risques sont en réalité minimes du fait du très haut degré de complexité d'un logiciel. L'entreprise éditrice du logiciel sera la seule à bien connaître le logiciel et son application. Un logiciel n'est pas uniquement constitué de connaissance codifiée (code source), son fonctionnement, son architecture et son utilisation reposent essentiellement

sur de la connaissance tacite. La connaissance tacite est très difficilement transmissible et son acquisition nécessite un coût très important. Nous pouvons raisonnablement penser que le coût d'acquisition de cette connaissance tacite est suffisamment élevé pour empêcher l'apparition de concurrents « parasites » qui utiliseraient le travail déjà réalisé pour créer un produit concurrent. Ce concurrent n'aurait pas à amortir le coût de développement du logiciel, il s'agirait bien sûr de concurrence déloyale. Mais du fait du coût élevé d'acquisition de la connaissance tacite, ce scénario noir pour la société éditrice, n'est pas envisageable.

Cependant, la société éditrice du logiciel ne peut pas pour autant bénéficier d'une véritable rente de monopole, puisqu'elle devra rester raisonnable dans ses prix de vente et continuer d'investir dans la Recherche et Développement. En effet, si le prix de vente est très élevé, le coût d'acquisition pour une société concurrente de la compétence nécessaire pour s'approprier le logiciel sera inférieure aux bénéfices qu'elle pourra escompter d'une telle démarche.

On voit que le logiciel libre propose un système d'incitation à l'innovation supérieur à celui basé sur les licences d'utilisation. Ce système d'innovation possède les avantages de la licence (protection de l'entreprise innovante pour lui permettre d'amortir les coûts de l'innovation) sans en avoir les inconvénients (création d'une rente de monopole avec toutes ces conséquences).

Cependant, cette logique ne peut pas s'appliquer à tous les types de logiciels. Il ne faut pas que ce soit un logiciel avec une forte diffusion (le coût d'acquisition des compétences serait vite amorti), surtout il faut que ce soit un logiciel dont la mise en place soit complexe et la programmation relativement simple. En effet, dans le cas de telles particularités le coût d'appropriation des compétences serait plus faible que le bénéfice potentiel que pourra en obtenir une forme concurrente. L'entreprise innovante sera désavantagée, et n'aurait aucun intérêt à innover.

Ce Business Model ne peut donc pas s'adapter aux logiciels libres grands publics ou très répandus comme Apache, Netscape, ou les distributions Linux, mais il est un modèle de croissance intéressant pour une entreprise éditrice de logiciels professionnels spécialisés.

b. IDX-PKI

IDX-PKI est la solution PKI (*Public Key Infrastructure*), développée par la société française IdealX²¹ spécialisée dans la vente de logiciel à façon et de services dans le domaine de l'Open Source

En résumant simplement une PKI est une solution capable d'assurer la sécurité des transactions à base de clé publique. Il s'agit d'un logiciel « à façon » car il doit être complètement adapté selon chaque entreprise.

Sur un marché en forte progression, la solution IDX-PKI répond complètement aux besoins de sécurité de l'entreprise : elle gère les autorisations d'accès ; garantit l'intégrité des données lors des échanges ; assure la confidentialité lors des transferts ; permet une authentification sûre des acteurs intervenant dans la transaction ; et, enfin, certifie le caractère non répudiable d'un message ou d'une opération.

Mais contrairement aux autres solutions PKI, IDX-PKI présente deux avantages principaux : l'utilisation d'une technologie Open Source offre une réduction considérable des coûts (entre cinq et vingt fois selon IdealX), et permet à la solution de s'adapter à n'importe quelle infrastructure. « *Nous maîtrisons le code, ce qui nous permet de développer le bon composant pour la bonne architecture. Pour les PME par exemple, il est possible de dimensionner la PKI en fonction du volume de ses transactions. Mais pour se faire une idée, le client peut télécharger l'infrastructure et la tester gratuitement avant de l'adopter définitivement* », explique Benoît Picaud, consultant Open Source chez IdealX.

L'éditeur ne demande aucune licence pour l'exploitation de sa PKI. « *Il n'y a aucun frais lié à la récupération et à l'exécution du programme. C'est assez nouveau pour un logiciel. Du coup, notre solution est sensiblement moins chère que les solutions à base d'autres logiciels. Cela permet aux entreprises d'investir dans d'autres postes budgétaires* », ajoute Benoît Picaud. En effet, la mise en place de PKI nécessite plusieurs phases d'adaptation, d'installation, de mise en production et de maintenance. C'est sur cette activité de prestations de services que la société a basé une partie de son modèle économique. « *Nous facturons pour le travail effectivement produit, mais nous savons*

²¹ <http://www.idealx.com>

que notre solution peut nous rendre révocable dès lors qu'une autre entreprise a toute latitude pour venir maintenir le code que nous avons écrit. Et le fait que nos concurrents travaillent sur la PKI les transforme aussitôt en collaborateurs. C'est le fait que nous jouons franc-jeu qui persuade nos clients de la pérennité de la solution », poursuit Benoît Picaud.

Disponible sous licence GPL (*Global Public Licence*), le produit est bien entendu conforme aux RFC (*Request For Comment* de l'IETF), garantissant ainsi l'interopérabilité avec les autres infrastructures PKI. En somme, IDX-PKI dispose de tous les atouts pour être utilisée dans un contexte industriel. Selon IdealX, le produit est capable de supporter de très vastes déploiements, allant au-delà du million d'utilisateurs.

6. Conclusion

Nous venons de présenter les cinq Business Models que l'on trouve dans le monde du logiciel libre.

- **Le modèle non Business**
- **Le modèle service et hardware**
- **Le modèle des distributions Linux**
- **Le modèle défensif**
- **Le modèle des logiciels « à façon »**

En étudiant ces Business Models nous nous sommes aperçus qu'il existait de fortes interactions entre ces modèles. Un Business Model du Libre ne peut être viable que s'il s'inscrit dans une dynamique globale. Cette dynamique des interactions entre ces différents Business Models est le fondement d'un modèle de croissance global du Libre. C'est ce modèle que nous allons désormais étudier.

IV. Modèle de croissance global

1. Système d'interaction entre les Business Models

Les cinq Business Models du logiciel libre que nous venons de définir et d'analyser, ne peuvent pas exister indépendamment les uns des autres, à l'exception peut être du *Modèle des logiciels « à façon »* qui suit une logique différente des quatre autres. Cette dynamique entre les différents Business Models du Libre est le fondement du modèle de croissance global du Libre.

Nous avons modélisé le système d'interaction entre les différents Business Model du Libre sur le schéma intitulé « *La Dynamique des Business Model du logiciel libre* ». (fig. 7)

a. Le modèle non Business : la clef de voûte du Libre

Le *modèle non Business* est au cœur de la dynamique des Business Models du libre. Le modèle de croissance global du Libre est le fruit de l'interaction des modèles *services et hardware*, *des distributions Linux*, *modèle défensif* et dans une moindre mesure du modèle *des logiciels « à façon »* avec le modèle *non business*. Pour les trois premiers modèles les projets de logiciels libres sont à la source de leur Business Model. En même temps, nous avons vu que le modèle *non business* ne serait pas viable sans le soutien des entreprises qui vivent du logiciel libre

Nous allons voir les interactions du modèle *non Business* avec les quatre autres Business Models du logiciel libre.

- *Modèle Service et hardware* : Ce modèle repose complètement sur les logiciels libres qui sont leur source de revenus, puisque les entreprises de services vivent grâce à la vente de services aux entreprises qui utilisent des logiciels libres. De même, pour les entreprises de hardware, elles vendent des solutions matérielles avec des logiciels libres préinstallés. Ces entreprises soutiennent les projets de logiciel libre soit par des dons soit en employant des développeurs qui travaillent à des projets Open Source.

- *Modèle des distributions Linux* : Le logiciel libre est aussi à la source des revenus des sociétés éditrices de distributions Linux que ce soit par la vente de « boîtes » ou par les offres de services informatiques. En contre partie les sociétés éditrices de distributions participent aussi à la viabilité des projets Open Source. Une partie de ses développeurs travaillent à améliorer le noyau de Linux, d'autres participent aux développements d'applications pour Linux.
- *Modèle défensif* : Le Libre est un système de production qui permet de donner une seconde vie à des logiciels propriétaires sur le déclin. En contrepartie de cette possible résurrection, le logiciel offre son code source à la communauté du Libre.
- *Modèle des logiciels « à façon »* : Les entreprises qui vivent de la conception de logiciels à façon développent principalement en interne et ne profitent donc pas de l'avantage du mode de production coopératif du Libre (le Bazar). Elles ne profitent pas directement des projets basés sur *le modèle non Business*, elles ont donc une place à part. Les sociétés éditrices de logiciels à façon n'interagissent pas avec l'ensemble des projets du Libre mais simplement avec leur projet Open Source. (Cependant, les sociétés éditrices de logiciels « à façon » ont aussi des activités de services, et elles interagissent alors comme une société de service)

b. Interaction entre les projets

Le modèle de développement global du Libre repose sur le réinvestissement par les sociétés qui vivent du logiciel libre d'une partie de leur bénéfice. Le Libre repose aussi sur des interactions entre les différents projets de logiciels libres.

Un logiciel libre ne peut pas s'imposer seul, son succès entraîne inévitablement le succès de nombreux autres logiciels libres. De même, les défauts d'un logiciel peuvent empêcher le développement de nombreux autres logiciels. Par exemple la réussite d'Apache, a entraîné le succès pour PHP, qui est un langage de programmation pour le Web et qui tourne sur les serveurs Apache. Linux par exemple a énormément de mal à

s'implanter comme système d'exploitation pour le grand public, car il n'existe pas de Suite bureautique comparable à celle de Microsoft. Par contre, le fait que Netscape rende Open Source son navigateur, a beaucoup aidé Linux qui possède désormais un navigateur de très haute qualité.

Encore une fois un logiciel libre ne peut pas espérer réussir tout seul, son succès passe nécessairement par un succès global du Libre.

c. Conclusion

La viabilité du modèle de croissance global du Libre repose sur les interactions entre le modèle non Business et les quatre autres Business Models du Libre. En effet, le modèle *non Business* permet la création de logiciels qui sont la source de revenus des entreprises qui gravitent autour du Libre. Ces entreprises, en investissant une partie de leur bénéfice dans les projets de logiciel libre assurent la pérennité de leurs projets.

En outre, il existe de nombreuses interactions entre les différents logiciels libres. Chaque logiciel libre est inscrit dans une continuité ascendante et descendante. Le succès d'un logiciel libre est étroitement lié au succès des autres.

La viabilité de chacun des Business Models est dépendante du succès du Libre en général. Le Libre est bien plus qu'un simple modèle de croissance, il est une approche radicalement différente de l'industrie logicielle. S'interroger sur la viabilité du modèle de croissance du logiciel libre revient à se demander si l'approche de l'industrie Open Source peut s'imposer sur l'approche Propriétaire.

2. Viabilité du modèle global de croissance du Libre

Nous venons de voir que le succès des différents Business Models reposait sur une réussite collective, car ces Business Models sont très interdépendants. Donc pour étudier la viabilité de ces Business Models il faut se pencher sur la viabilité du logiciel libre en général.

Les chances de succès du modèle global du Libre sont différentes selon les contextes. On peut distinguer deux marchés très différents dans l'industrie informatique celui des entreprises (Business to Business) et celui du grand public (Business to Consumer).

a. Le marché des entreprises.

Le marché du logiciel d'entreprise est particulièrement propice au logiciel libre. D'ailleurs certains logiciels libres sont déjà leaders sur leur marché. C'est le cas par exemple d'Apache avec plus 65% des parts de marché des serveurs Internet. Plus de 30% des serveurs Internet des entreprises tournent sous Linux. Nous avons vu aussi que le modèle des *logiciels « à façon »* semble très prometteur.

D'ailleurs les grands acteurs de l'industrie de l'informatique d'entreprise ont fait le pari de l'Open Source. IBM en particulier investit énormément dans Linux, tout comme Sun qui, après avoir lutté pour imposer son propre système Unix, se tourne désormais vers Linux.

Le marché de l'entreprise est un marché où le modèle global de croissance du Libre peut le mieux s'exprimer. En effet, les entreprises ont besoin de systèmes complexes, stables, pérennes, et adaptables, qui sont autant de qualités propres aux logiciels libres.

De plus, on a vu que le modèle de croissance du Libre repose sur des entreprises qui vivent grâce aux logiciels libres et qui réinvestissent dans des projets Open Source. La source principale de revenu pour ces entreprises est la vente de services informatiques. Le marché de l'informatique d'entreprise est un marché où les services informatiques comptent beaucoup. C'est pour cette raison que le Libre semble être une forme d'organisation de l'industrie logicielle particulièrement adaptée pour le marché de l'informatique d'entreprise.

Il faut noter une exception toute fois, c'est celle du marché des stations de travail où le Libre est confronté à des obstacles très importants pour pouvoir s'imposer. Ces obstacles sont de même nature que ceux qu'il rencontre sur le marché grand public, c'est-à-dire la difficulté de s'imposer face à un monopole.

b. Le marché grand public

Le marché grand public est moins propice à un développement du Libre que le marché des entreprises. Il y a à cela deux raisons : un environnement défavorable et un manque d'adaptation au marché grand public du modèle de croissance global du Libre tel que nous venons de le définir.

Sur le marché grand public, il est très difficile de lutter contre le quasi-monopole du système d'exploitation Windows de Microsoft. Car le fort « effet de club » qui caractérise l'économie du logiciel, induit la mise en place de situations de lock-in contre lesquelles il est quasi-impossible de lutter. En outre, depuis la sortie de sa dernière version (Windows XP), le système d'exploitation de Microsoft est devenu stable et très fonctionnel, ce qui rend encore plus difficile l'émergence d'un système concurrent, qu'il soit libre ou propriétaire.

De plus, le Libre possède une particularité qui aurait pu le faire percer malgré tout sur ce marché, c'est la possibilité de copier et de distribuer gratuitement les logiciels libres puisqu'ils ne sont pas soumis à des licences d'utilisation. Cette gratuité pourrait être un avantage décisif face au prix très élevé des licences Microsoft. Cependant, ce scénario n'est pour l'instant pas envisageable : d'abord parce que la majorité des ordinateurs PC sont vendus avec Windows préinstallé, le consommateur n'a pas conscience du prix de la licence ; d'autre part la tolérance inavouée de Microsoft pour le piratage lui permet de continuer d'imposer ses standards.

Mais les difficultés du Libre pour s'imposer sur le marché grand public ne sont pas seulement dues à l'hostilité de l'environnement mais aussi aux faiblesses propres au Libre. D'abord, il est très difficile de trouver pour les entreprises un Business Model qui puisse s'adapter au marché grand public.

Nous avons vu que les distributions Linux ont d'abord essayé un Business Model appuyé sur la vente de « boîtes », mais que ce modèle avait de nombreuses limites, et aujourd'hui elles se retournent vers un modèle de services, qui semble être le Business Model, le plus efficace pour le Libre. Cependant, ce modèle du service n'est pas adapté au marché grand public. En effet, les logiciels grands publics, pour s'imposer, doivent être simples d'utilisation ; ce qui réduit sensiblement les possibilités de service d'installation ou de formation. Les entreprises qui se tourneraient vers le marché grand public ne pourraient espérer qu'un faible chiffre d'affaire donc leur participation financière aux projets Open Source serait relativement faible. Mais en même temps ces entreprises devraient utiliser de nombreux logiciels Open Sources. Il risquerait d'y avoir très rapidement une rupture de l'équilibre entre l'apport financier des entreprises du Libre et les coûts d'un développement à très grande échelle même dans le cadre du Bazar. Le modèle actuel du Libre qui repose sur l'interaction du modèle *non Business*, avec les

autres Business Model du Libre, ne pourrait plus fonctionner. La solution pourrait reposer sur des voies alternatives de financement du modèle *non Business*.

Tous ces éléments font que le logiciel libre aura beaucoup de difficultés à s'imposer sur le marché grand public, cependant l'avenir n'est pas écrit, et la situation pourrait changer.

D'abord, les logiciels libres pourraient profiter des développements, qu'ils vont connaître pour le marché de l'entreprise pour améliorer encore leurs performances. Le propre des logiciels libres est qu'ils évoluent très vite, il est donc possible qu'ils continuent de se développer et de s'améliorer. S'ils continuent de progresser plus rapidement que leurs concurrents propriétaires, le différentiel de qualité pourrait leur permettre de s'imposer malgré tout sur le marché grand public.

En outre, l'environnement actuel très défavorable pourrait évoluer en faveur du Libre. Microsoft affiche une politique anti-piratage très marquée. Si la firme de Redmond arrive à lutter réellement efficacement contre le piratage, il existerait une réelle opportunité pour les logiciels libres, en particulier de Linux de s'imposer sur le marché grand public. De plus, Microsoft n'est toujours pas à l'abri de mesures prises par la justice pour situation de monopole. Un démantèlement de Microsoft favoriserait l'émergence de logiciels concurrents et en particulier de logiciels Open Source. En outre, le Libre pourrait aussi être favorisé par une politique volontariste des Etats qui ne veulent pas que leur système d'information soit dépendant d'une firme privée étrangère. C'est le cas en particulier de la Chine qui investit énormément dans les logiciels Open Sources, comme le montre l'exemple symbolique du développement de la distribution Red Flag Linux. En Europe aussi les gouvernements commencent à s'intéresser aux logiciels libres pour les installer dans les administrations publics et l'éducation nationale.

Si l'environnement pourrait devenir plus favorable au Libre, il n'en demeure pas moins que le modèle de croissance du Libre semble bien mal adapté au marché grand public. Une piste de solution a été proposée par Foray et Zimmermann (2002). Le mode de production coopérative du Libre (Bazar) est assez comparable à celui de la recherche avec un système d'incitation basée par la reconnaissance par ses pairs. Dans ce cas ne pourrions nous pas imaginer, un système qui fonctionnerait comme la recherche publique, c'est-à-dire avec des développeurs payés par l'état. Le modèle global de

croissance du Libre en serait modifié mais cela permettrait d'assurer la viabilité de ce modèle en particulier sur le marché grand public.

c. Conclusion

Nous avons montré que la viabilité des différents Business Models du logiciel libre, étaient dépendants de la réussite des autres Business Models. Cette interdépendance constituait le modèle global de croissance du Libre. Donc s'interroger sur la viabilité des différents modèles revenait à s'interroger sur la viabilité du modèle global de croissance du Libre.

Nous avons montré que ce modèle de croissance du Libre était particulièrement adapté au marché des entreprises. Par contre dans la situation actuelle, le Libre aura de très grandes difficultés d'apparaître une organisation crédible de l'industrie du logiciel.

Conclusion

Conclusion

Générale

Conclusion

La problématique de ce mémoire était de « *comprendre les différents types de Business Models du logiciel libre, pour essayer d'appréhender la viabilité de cette nouvelle approche de l'industrie logicielle* ». Pour répondre à cette problématique nous avons procédé en trois étapes.

La première partie de ce mémoire, nous a permis de mieux comprendre les aspects techniques, juridiques, et historiques du logiciel libre. Nous avons ainsi montré que la caractéristique majeure d'un point de vue technique du logiciel libre est qu'il est distribué avec son code source. Cette ouverture du code source fait que le logiciel libre est caractérisé par 4 niveaux de libertés (liberté d'étudier, d'exécuter, de redistribuer et d'améliorer le code source). Ces libertés sont assurées par un cadre juridique précis, qui consiste à détourner le droit d'auteur (copyleft).

Dans une deuxième partie nous avons montré que les libertés associées au logiciel Open Source permettent d'envisager un mode de production coopératif. Ce mode de production (le Bazar) s'oppose à celui du logiciel propriétaire (la Cathédrale) et repose sur un double système d'incitation des développeurs (« *career concern incentive* » ; « *the ego gratification incentive* »). Puis nous avons vu que, grâce à ce mode de production coopératif, les logiciels libres possèdent de nombreuses qualités par rapport aux logiciels propriétaires.

Ces deux premières parties nous ont permis de voir les aspects généraux du logiciel libre et de son économie, dans la troisième partie nous avons traité ce qui constitue le cœur et l'originalité de ce mémoire : l'étude des Business Models du logiciel libre.

Nous avons d'abord, défini la notion de Business Model et montré que sa meilleure traduction était celle de modèle de croissance. Puis nous avons montré que le point commun de tous les modèles de croissances du Libre était de reposer sur des faibles coûts : faibles coût de productions (Bazar) et de marketing (marketing viral). Ensuite nous avons défini et analysé les cinq types de Business Model que l'on trouve dans l'industrie du logiciel libre :

- *Le modèle non Business*
- *Le modèle service et hardware*
- *Le modèle des distributions Linux*

- *Le modèle défensif*
- *Le modèle des logiciels « à façon »*

Nous avons alors montré que ces Business Models n'étaient pas indépendants mais s'articulaient autour du modèle *non Business.*, nous avons modélisé ces interactions par un graphique (Fig.7). Puis nous avons montré que cette dynamique entre les différents Business Models correspondait au *modèle global de croissance du Libre.* Enfin nous avons vu que ce modèle global de croissance était particulièrement efficace sur le marché de l'informatique d'entreprise. Par contre sa viabilité sur le marché de l'informatique grand public était plus problématique.

Ce mémoire nous a montré que les Business Models du logiciel libre n'étaient pas simplement de nouveaux modes de croissance, mais qu'ils s'inscrivaient dans une logique plus vaste qui correspond à une nouvelle forme d'organisation de l'industrie logicielle.

Cette industrie a déjà connu deux trajectoires technologiques : la première, jusqu'au début des années 1980 où le logiciel était associé au hardware ; et la deuxième depuis les années 1980, celle du logiciel propriétaire. Le libre est-il une nouvelle trajectoire technologique capable de supplanter le propriétaire ? C'est possible mais pas certain. En effet, si ce mémoire nous a montré que le Libre avait de atouts indéniables pour s'imposer, en particulier sur le marché de l'informatique d'entreprise, la généralisation de cette organisation de l'industrie du logiciel soulevait des problématiques nouvelles. En particulier, sur la viabilité du mode coopératif de production (le Bazar) : saurait-il s'adapter à une généralisation ?

Ces nouvelles problématiques sont riches et devraient donner lieu à d'autres travaux de recherche.

Bibliographies

Alcove, Le site Web de la société Alcove <http://www.alcove.fr>

Apache, Le site Web officiel de La Apache Software Foundation <http://www.apache.org>

Arrow K J. (1962), « *Economic Welfare and allocation of Ressources to Invention* » dans NELSON (ed), Princetown University Press

Brousseau E (1993), « *L'économie des contrats : technologies de l'information et coordination interentreprise* » P.U.F.

David, P. et Foray D. (1995), « *Accessing and expanding the science and technology knowledge base* », *STI Review*, n°16, OCDE, Paris

Callon M, (1991), « *Réseaux technico-économiques et irréversibilités* » Boyer, Chavance Godard

Carcenac T. (2001), *Pour une administration électronique citoyenne*
<http://www.internet.gouv.fr/francais/textesref/rapcarcenac/sommaire.htm>

Coase R. H. (1959), « *The FCC* », *Journal of Law and Economics*, n°2, pp1-40

Cybersource (2002), *Windows vs linux : TCO comparaison*
http://www.cyber.com.au/cyber/about/linux_vs_windows_tco_comparison.pdf

Foray D., Zimmermann J.B. (2002), *L'économie du logiciel libre : Organisation coopérative et incitation à l'innovation*, *Revue économique*, vol. 52, p77-93

FSF, Le site Web officiel de la Free Software Foundation <http://www.fsf.org>

Kelty C. (2001), *Free Software/Free Science*,
http://www.firstmonday.org/issues/issue6_12/kelty/index.html

IdealX, Le site Web de la société IdealX <http://www.idealx.com>

Jullien N. (2001), thèse de doctorat, « *Impact du logiciel libre sur l'industrie informatique* » Université de Bretagne Occidentale

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

Lévy P. (1992) « *de la programmation considérée comme un des beaux-arts* », La Découverte

Linuxfr, Le site de référence de la communauté du Libre Francophone

<http://www.linuxfr.org>

Maître B., Aladjidi G. (1999) *Les Business Models de la nouvelle économie*, Dunod.

Mandrake, Le site Web de la distribution Linux-Mandrake <http://linuxmandrake.fr>

Mowery D. C. (1996), « *The International Computer Software Industry, A comparative Study of Industry Evolution and Structure* », Oxford University Press

Open Source, Le site Web du mouvement open source <http://www.opensource.org>

Quah (1999), *The weightless economy in economic development*", London School of Economics Economic Department, Londres

Rastetter Y. (2002), *Le logiciel libre dans les entreprises*, Lavoisier

Raymond E.S. (1998), *La Cathédrale et le Bazar*, traduit par Blondeel S.,

http://www.lifl.fr/~blondeel/traduc/Cathedral-bazaar/Main_file.html

Samuelson P. A., (1954) ; « *The Pure Theory of Public Expenditure* » *I Review of Economics and Statistics*, n°36, pp 387-389

Slashdot, Le site Web de référence de la communauté du libre, <http://www slashdot.org>

Torwalds L. (2001), *Pourquoi le modèle open source est sensé*, chapitre du livre *Il était une fois linux*, Osman Eyrolles Multimédia, p264-276

Zimmermann J-B (1995), « *L'industrie du logiciel : de la protection à la normalisation* » dans Basle, Dufourt, Héraud, Perrin, *Changement institutionnel et changements technologiques Evaluations, droits de propriété intellectuelle, système national d'innovation*, CNRS Edition.