Laure MUSELLI

CEPN (Centre d'Economie de l'Université Paris Nord)
99, avenue Jean-Baptiste Clément,
93430 Villetaneuse.

Téléphone professionnel :
01 49 40 20 80
01 60 76 44 17
Fax :
01 49 40 33 34

Adresses électroniques :
Laure.Muselli@int-evry.fr
laure_muselli@yahoo.fr

13ème conférence de l'AIMS. Normandie. Vallée de Seine 2, 3 et 4 juin 2004

« LICENCES INFORMATIQUES ET MODELES D'AFFAIRES OPEN SOURCE »

Mots-clés :

Open Source, Logiciels, Licences, Valorisation, Business Model, Coopération, Diffusion.

Cet article traite de la question de l'émergence du mouvement Open Source dans l'industrie du logiciel. Ce mouvement est caractérisé par l'utilisation croissante d'un nouveau type de licence informatique : les licences Open Source, autorisant, souvent gratuitement, la libre utilisation, modification et distribution des logiciels. Si l'intérêt de ces licences pour la sphère non-marchande - notamment les universités et les laboratoires de recherche publique - semble évident, il n'en va pas de même pour la sphère marchande.

Dans cette perspective, notre objectif consiste à montrer dans quelles conditions l'utilisation de ce type de licence peut s'avérer compatible avec la sphère marchande, et notamment les objectifs de valorisation inhérents à toute entreprise commerciale d'édition de logiciels. En corollaire, nous mettrons en exergue l'importance du choix de la licence, car ce point constitue aujourd'hui un enjeu majeur, notamment parce qu'il ne peut se trouver négligé si l'on veut assurer la pérennité du modèle économique de l'entreprise. Nous insisterons également sur les enjeux stratégiques de l'Open Source, en plaçant plus largement la problématique du choix de licence et de business model dans un contexte où l'existence d'une forte coopération influence la diffusion et la valorisation des innovations.

Après avoir clarifié la notion de licence informatique et proposé une typologie des différentes licences open source existantes, nous présenterons donc les modèles d'affaires ou « business models » susceptibles d'être utilisés par des éditeurs commerciaux de logiciels open source. Pour chacun de ces «business models », nous expliquerons, en nous appuyant sur des exemples concrets, de quelle manière ils permettent aux éditeurs commerciaux de logiciels de valoriser leur activité, mais également quels peuvent être leurs intérêts et limites, ainsi que leurs conditions d'efficacité.

Licences informatiques et modèles d'affaires Open Source

L'industrie du logiciel assiste, ces dernières années, à l'apparition d'un phénomène nouveau : la démocratisation des logiciels dits «libres » ou «Open Source ». Ces concepts répondent, non à une idée de logiciels gratuits, mais plutôt à celle de logiciels ouverts : ceci se matérialise par l'application d'une licence particulière qui autorise l'accès au code source du programme et garantit les libertés d'utilisation, modification et distribution du logiciel par l'utilisateur. Les licences libres et Open Source s'opposent en cela aux licences propriétaires, traditionnellement utilisées par les éditeurs de logiciels commerciaux, qui interdisent l'accès au code source et restreignent les libertés d'utilisation, de modification et de distribution du logiciel par les utilisateurs, tout en leur imposant généralement une redevance. La non-divulgation de ces sources constitue d'ailleurs, pour les éditeurs de logiciels propriétaires, un moyen de protection, au même titre que le droit d'auteur[i].

Le Libre et l'Open Source constituaient, à l'origine, des logiques de protection intellectuelle développées pour la sphère non marchande, notamment le monde universitaire de la recherche ou les organisations non commerciales, menant des projets qui visent à développer des alternatives libres concurrentes de produits propriétaires (comme le système d'exploitation Linux, concurrent direct de Microsoft Windows). Elles sont aujourd'hui adoptées par un nombre croissant d'éditeurs commerciaux de logiciels, qui les préfèrent parfois aux licences propriétaires traditionnelles. On ne peut assister à ce phénomène sans se demander quelles peuvent être les raisons de son apparition Ainsi, au-delà des seules considérations philosophiques souvent associées au nouveau phénomène des logiciels libres, les travaux concernant l'Open Source ont, dans un premier temps, tenté de comprendre le fonctionnement même de ce mouvement, tant l'idée d'une contribution gratuite aux programmes Open Source semblait difficile à expliquer (Prasad 2001 et Raymond, 2001). Cela a donné naissance à deux types de travaux : d'une part ceux concentrés sur les motivations des individus à coopérer aux logiciels libres, (Lerner et Tirole, 2002a ; Dalle et David, 2003), et d'autre part ceux traitant des processus de développement et les aspects manageriaux et organisationnels d'un projet Open Source (Mockus, Fielding et Herbsleb, 2000 ; Von Hippel, 2001 ; Steinmueller et Garcia, 2003). Aujourd'hui, certaines recherches

commencent à se focaliser sur les raisons poussant les entreprises informatiques elles-mêmes à prendre part au mouvement open source (Bonaccorsi et Rossi, 2003 ; Jullien, 2003).

Comme le montre Muselli (2002), les motivations stratégiques des éditeurs optant pour un modèle Open Source, peuvent consister, dans un premier temps, en certains avantages procurés par l'Open Source en termes de coopération (de la communauté de programmeurs), ainsi que de diffusion, en vue de l' imposition d'un produit comme standard de fait. Toutefois, la problématique de la valorisation ne peut être négligée, le but de toute entreprise commerciale consistant en effet à générer des revenus nécessaires à son fonctionnement et à réaliser des profits. L'étude des «modèles Open Source » nous paraît donc constituer une question primordiale, qui se rattache plus largement aux questionnements traditionnels de la gestion sur la diffusion des innovations (Rogers, 1995) et leur valorisation.

Nous entendons, par « modèle Open Source », le mécanisme par lequel une entreprise tente de créer une source de revenus, tout en utilisant, pour un ou plusieurs de ses produits, une licence Open Source. Quelques travaux récents vont d'ailleurs dans ce sens. Raymond (1999) donne un aperçu des différentes sources de valorisation susceptibles de pérenniser les modèles open source. Hawkins (2002) présente un modèle mathématique des bénéfices attendus par un producteur de logiciel open source, fonction de ses coûts (en termes de pertes résultant de la divulgation de son code source et d'économies liées à la coopération d'une communauté de programmeurs) et de ses revenus (constitués des ventes de hardware ou de services). Pal et Madammohan (2002) établissent des règles de réussite d'un modèle Open Source, sans toutefois développer l'impact des licences utilisées. Enfin, Välimaki (2003) et Van Wendel de Joode, de Bruijn et Van Eeten (2003) présentent le système de double licence, un des modèles de valorisation que nous expliciterons plus en détails.

Cet article possède un double objectif : tout d'abord, clarifier la notion générale de « modèle Open Source » pour les éditeurs de logiciels commerciaux[ii], en montrant qu'il n'existe pas un modèle unique, mais une pluralité de modèles basés sur trois formes génériques; ensuite, et c'est là l'originalité de notre travail, mettre en exergue l'importance du choix de la licence informatique dans le bon fonctionnement de chacun de ces modèles open source.. Notre article sera organisé comme suit : dans une première partie, après avoir clarifié quelques concepts, nous établirons une typologie des licences de logiciels, indispensable à la compréhension de notre exposé. Nous présenterons alors, dans un deuxième temps, les trois modèles d'affaires génériques utilisables par les éditeurs de logiciels, dont nous mettrons en avant les intérêts, limites et risques, ainsi que les conditions requises pour qu'ils soient susceptibles de fonctionner au mieux.

1. Quelques concepts

1.1. La nature de la licence informatique

Comme nous l'avons évoqué en introduction, le système de protection intellectuelle auquel sont soumis les logiciels est le droit d'auteur[iii]. Ainsi, tout individu mettant au point un logiciel, peut bénéficier sur celui-ci, sous condition d'originalité, d'un droit de propriété intellectuelle qui est le droit d'auteur. La licence n'est autre que la matérialisation de l'utilisation que va faire l'entreprise de son droit de propriété intellectuelle sur ce logiciel. Elle consiste en un contrat avec l'utilisateur, donnant le droit à celui-ci d'utiliser le logiciel, en lui imposant en contrepartie des obligations plus ou moins restrictives ainsi qu'une redevance plus ou moins élevée. Ainsi, même les licences dites « libres » sont un ensemble particulier de clauses combinées avec le droit d'auteur : celles-ci font en sorte de garantir les libertés octroyées à l'utilisateur, par opposition à celles d'une licence propriétaire, qui les restreignent. Ce n'est donc pas le copyright, droit automatique donné à tout auteur, qui détermine le caractère libre ou pas du logiciel, mais bien la licence, selon que ses clauses limiteront ou préserveront les libertés.

1.2. Les notions de licences « libres » et « open source »

Les significations des appellations « licence libre » et « licence open source » diffèrent peu dans la pratique, mais répondent à des critères définis par deux associations différentes : La FSF (Free Software Foundation) pour les licences libres et l'OSI (Open Source Initiative), pour les licences open source.

Les caractéristiques d'une **licence « libre »** se résument en trois points :

- la liberté d'utiliser le logiciel, c'est-à-dire d'exécuter le programme,
- la liberté d'accéder au code source – un ensemble d'instructions écrites dans un langage de programmation informatique compréhensible par les humains et permettant d'obtenir un programme pour un ordinateur. [Wikipedia, l'encyclopédie du libre]-, quelle qu'en soit la finalité (étudier le fonctionnement du programme, adapter le logiciel à ses besoins personnels, améliorer le programme en en corrigeant les erreurs),
- la liberté de distribuer des copies du logiciel, par quelque voie que ce soit, à titre gratuit ou pas**.**

Une **licence «open source »** doit satisfaire une dizaine de critères, se voulant moins stricts que ceux du libre, de façon à permettre l'utilisation de telles licences par les entreprises commerciales[iv].

Dans la suite de cet article, nous utiliserons indifféremment l'un et l'autre de ces deux termes.

1.3. Une typologie des licences Libres et Open Source

On a souvent tendance à restreindre la notion de licence libre à celle de GPL (Gnu Public License), licence libre régissant notamment le système d'exploitation Linux, et à considérer que cette dernière est la seule . C'est pourtant loin d'être le cas si l'on se réfère à la liste publiée par l'OSI, qui recense les licences open source. On n'y trouve pas moins d'une trentaine de licences open source, qui répondent toutes aux différents critères de liberté, mais qui vont différer par certaines clauses. Lee (1999) et Van Wendel, De Bruijne et Van Eeten (2003) présentent, dans leurs travaux, les caractéristiques de quelques unes de ces licences. Le but de cette partie consiste, non pas à réaliser une présentation exhaustive des licences Open Source, mais plutôt à en proposer une classification simple, indispensable pour comprendre les enjeux des modèles de valorisation.

1.3.1. Les licences libres copyleftées

Le copyleft est un concept qui a été créé par Richard Stallman, fondateur en 1980 de la Free Software Foundation, dans le cadre du projet GNU, projet visant à créer des logiciels libres (que ce soient un système d'exploitation, mais aussi des programmes fondamentaux tels que des débogueurs, des compilateurs, ou encore, plus récemment, des outils bureautiques). L'élaboration du concept de copyleft avait alors pour but d'offrir un outil de protection alternatif à la licence non copyleftée, pour les logiciels ouverts, permettant d'éviter l'appropriation par des entreprises commerciales des logiciels développés dans un environnement de recherche.

Concrètement, mettre un logiciel sous copyleft consiste à coupler le copyright à une licence incluant une clause particulière interdisant de sous-licencier le programme. Ainsi, tout utilisateur a la permission d'exécuter, de modifier ou de distribuer librement le logiciel, dans la mesure où il transmettra ces mêmes droits aux utilisateurs de toutes les copies ou travaux dérivés dudit logiciel, sans y ajouter quelque restriction que ce soit. Tout utilisateur ayant modifié le code source d'un logiciel sous licence œpyleftée, est donc obligé de faire du produit obtenu un logiciel placé sous cette même licence. Par conséquent, ceci signifie, que

l'appropriation du logiciel par une entreprise devient impossible. Ces licences copyleftées peuvent être classées en deux catégories : les licences contaminantes et les licences persistantes.

1.3.1.1. Les licences contaminantes

Les licences copyleftées contaminantes imposent l'obligation de placer sous la même licence non seulement tout logiciel modifié, mais également tout programme incluant ce logiciel dans un produit dérivé. Ainsi, un produit incluant un module placé sous licence copyleftée contaminante ne peut être placé sous licence propriétaire.

La GPL (Gnu Public License) est la licence copyleftée contaminante la plus connue, mais également la plus utilisée. Elle n'est cependant pas la seule à appartenir à cette catégorie, puisque l'on peut citer également la MPL (Mozilla Public License).

1.3.1.2. Les licences persistantes

Les licences libres copyleftées persistantes possèdent les caractéristiques des licences libres ou open source, dans la mesure où elles contiennent les clauses autorisant les libres utilisation, copie, modification et distribution. Elles diffèrent cependant des licences copyleftées contaminantes, car elles comportent une clause autorisant les utilisateurs à mélanger le logiciel à un logiciel non libre et à placer sous licence propriétaire les produits dérivés ainsi développés, à condition que le module libre utilisé reste sous licence libre. En pratique, comme les licences contaminantes, les licences persistantes interdisent l'appropriation du logiciel et sa mise sous licence propriétaire. En revanche, à leur différence, elles autorisent l'intégration du module copylefté dans un produit dérivé propriétaire.

La licence LGPL (Lesser GPL) est l'exemple le plus connu de licence copyleftée persistante. A l'origine, créée spécialement par la FSF pour les bibliothèques logicielles, dans le but d'encourager leur diffusion, cette licence ne leur est pas exclusivement réservée et peut être appliquée de la même façon à d'autres types de logiciels ou programmes.

1.3.2. Les licences libres non copyleftées.

Cette catégorie de licence correspond à des licences libres, mais possédant la caractéristique d'être appropriables, du fait de l'absence de copyleft (c'est-à-dire de clauses instaurant une persistance).

Elles possèdent en commun les clauses suivantes : toute personne obtenant une copie du logiciel se voit attribuer les droits illimités de l'**utiliser**, le **copier**, le **modifier**, le

fusionner [avec d'autres logiciels, quelle que soit la licence qui les régit], le **publier**, le **distribuer**, le **sous-licencier** – ce qui signifie que tout utilisateur peut légalement décider de placer sa copie ou ses travaux dérivés sous une licence différente, quelle qu'elle soit - et/ou d'en **vendre des copies**, et de permettre aux personnes auxquelles elle les distribue de faire de même. Le droit donné aux utilisateurs de sous-licencier ce type de logiciel lui donne une caractéristique particulière : il permet à toute entreprise de s'approprier celui-ci en le plaçant sous sa propre licence propriétaire, et de limiter ainsi les droits des utilisateurs de cette nouvelle version du produit. Cependant, la version libre du programme d'origine reste toujours disponible, laissant la possibilité aux individus d'utiliser la version de leur choix, libre ou propriétaire. Le cas d'Unix est un bon exemple de ce type de pratique : ce système d'exploitation développé à l'université de Berkeley et placé sous licence BSD, fut approprié par de nombreuses entreprises, chacune en ayant développé sa version propriétaire propre : Solaris pour Sun Microsystems, IBM AIX pour IBM, ou encore HPUX pour Hewlett Packard.

Les licences Xfree86, X Consortium, BSD, MIT, X11, BSD modifiée et Apache appartiennent à cette catégorie de logiciels dits non copyleftés. Ce sont certaines clauses, notamment relatives à l'utilisation du nom du détenteur du copyright pour promouvoir la vente ou l'utilisation du logiciel, qui les différencient.

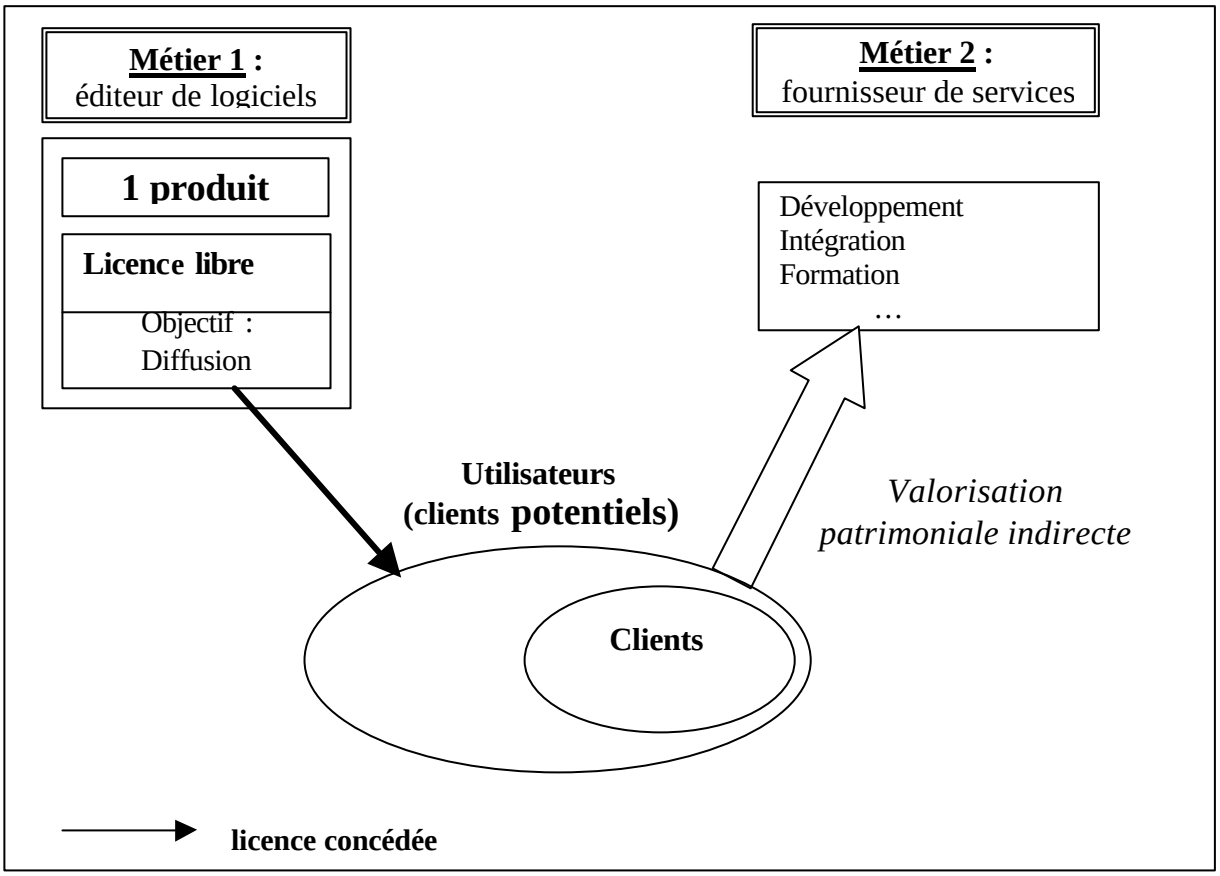
Tableau récapitulatif des caractéristiques des différents types de licences Open Source :

		Caractéristiques	
		Appropriabilité	Contamination
Type de licence	Copyleftée persistante	Non	Non
	Copyleftée contaminante	Non	Oui
	Non copyleftée	Oui	Non

Source : Laure Museli

2. Trois modèles de valorisation de l'Open Source

2.1. Valorisation indirecte : du métier d'éditeur au métier de fournisseur de services



Source : Laure Muselli

2.1.1. Présentation du modèle

Ce modèle s'appuie sur **un type de valorisation que nous qualifierons d'indirect**. Ici, le logiciel édité est placé sous une licence Open Source généralement gratuite et permettant aux utilisateurs de l'utiliser, le copier, le modifier et le redistribuer librement. L'entreprise ne tire donc ici pas directement de profit de l'utilisation de son produit à travers l'exploitation de sa licence, puisqu'elle concède ainsi un droit d'utilisation gratuit. La source de valorisation indispensable est donc à chercher ailleurs : elle va se faire par le biais de la prestation de services aux entreprises utilisatrices du logiciel en question. En effet, le fait que le logiciel ait été développé par l'éditeur lui-même confère à celui-ci des compétences et une

légitimité particulières à fournir des services tels que le développement, mais aussi l'intégration ou la formation, deux activités en forte croissance actuellement. On parlera donc de valorisation indirecte, dans la mesure où la rémunération proprement dite n'est pas conditionnée par les clauses de la licence mais, comme nous l'avons vu, issue d'une activité de service, (où l'exploitation des droits d'auteur au travers de la licence n'est plus de mise) et non plus d'une activité d'édition de logiciels. Il paraît important de bien considérer que ce modèle économique implique un changement d'activité de la part de l'entreprise, qui passe du métier d'éditeur de logiciel à l'origine, à celui de fournisseur de services basés sur les logiciels libres (SSLL)[v].

La société Open Cascade a décidé de suivre ce modèle de valorisation pour son produit de CAO (Conception Assistée par Ordinateur). Tout d'abord développé sous une licence commerciale, cet outil ne possédait qu'une part de marché réduite. Décision fut prise d'adopter une licence incitative de type Open Source (assimilable à une licence LGPL), de façon à démocratiser son utilisation. Une fois la base d'utilisateurs suffisamment développée, Open Cascade a pu ainsi rechercher une source de valorisation indirecte, non plus dans la vente de licences comme auparavant, mais dans la signature de contrats de fourniture de services complémentaires aux entreprises utilisatrices de son produit de CAO.

La société IdealX utilise également ce modèle, dans la mesure où elle a mis au point plusieurs solutions, comme par exemple une infrastructure de PKI open source, IDX PKI, pour les applications sécurisées, les transactions et le commerce électroniques. Elle ne valorise pas son activité directement, mais indirectement, grâce à la fourniture de services aux entreprises et administrations désireuses d'utiliser une de ces infrastructure.

2.1.2. Intérêt stratégique

L'intérêt stratégique de ce type de modèle se situe aussi bien dans la coopération de la communauté de programmeurs que dans la diffusion du produit.

Concernant la composante **« coopération »**, l'ouverture du logiciel autorise et encourage la création d'une communauté de programmeurs travaillant sur l'amélioration et la correction de bogues du logiciel, ce qui a pour conséquence d'en améliorer la qualité et les performances. Cette coopération ne concerne évidemment pas l'ensemble des utilisateurs, mais seulement une partie d'entre eux, capables de comprendre le code et de suggérer des modifications[vi]. Même si la gestion de cette communauté et de ses apports n'est pas sans coût pour l'éditeur, cette coopération source d'économie de R&D reste très bénéfique pour l'entreprise. Dans le cas d'Open Cascade, cette composante de coopération est présente, dans

la mesure où un site Internet est dédié à la gestion d'une communauté de programmeurs, qui proposent des modifications et améliorations susceptibles d'être intégrées aux versions suivantes du produit. Mais cette composante de coopération est plus importante encore dans le cas d'IdealX, qui assoit son image de marque sur les nombreux projets dont elle est à l'initiative, permettant mutualisation et approche contributive. C'est la crédibilité véhiculée par cet effort de mutualisation et de coopération, sources de qualité, qui permet à l'entreprise de se valoriser par son activité de fourniture de services autour de ces produits.

Pour ce qui est de la composante **« diffusion »**, on peut parler de composante stratégique essentielle : outre l'avantage pour l'utilisateur de pouvoir se procurer le produit gratuitement, l'utilisation d'une licence open source introduit une certaine transparence, permet au logiciel d'être testé, contribue, grâce à la communauté de programmeurs, à une meilleure qualité et confère finalement une certaine notoriété au produit. Grâce à ce choix, l'entreprise incite les utilisateurs à adopter son logiciel et démocratise l'utilisation de son produit. Une fois cette large diffusion réussie et la notoriété du produit accrue, l'entreprise peut alors espérer se rémunérer indirectement en proposant des services aux utilisateurs, comme l'a fait Open Cascade. Cette composante «diffusion » se place dans la logique du modèle de diffusion de nouveaux produits de Rogers (1995) : les membres de la communauté de développeurs tiennent ici le rôle d'« early adopters », qui déterminent l'adoption éventuelle du logiciel libre, de par leur capacité à tester le produit et à devenir des prescripteurs pour les autres catégories d'utilisateurs.

Finalement, un cercle vertueux se forme, car la coopération améliore la qualité des produits, ce qui entraîne la diffusion de ces derniers et augmente ainsi les possibilités de coopération.

2.1.3. Limites et risques

Si ce type de modèle paraît a priori parfaitement viable, l'entreprise doit cependant s'assurer de posséder une clientèle suffisamment importante pour générer un montant de revenus acceptable. Pour cela, elle a la possibilité de jouer sur deux variables : d'une part la taille de son marché et d'autre part le taux de transformation des utilisateurs (clients potentiels) en clients. L'utilisation de ce modèle pourra conduire à un succès seulement dans deux cas de figure : le premier est celui d'un marché de petite taille (de type marché de niche), mais où le taux de transformation des utilisateurs en clients s'avère élevé ; le deuxième est celui d'un taux de transformation plus faible, mais sur un marché plus vaste.

2.1.3.1. *Taille du marché potentiel*

La première limite du modèle open source de valorisation indirecte concerne la taille du marché potentiel. Celle-ci représente le nombre d'utilisateurs du logiciel open source, c'est-à-dire le nombre de clients potentiels, susceptibles d'effectuer une demande de services payants à l'éditeur / fournisseur de services. Plus ce marché est grand, plus l'entreprise aura la possibilité de toucher un nombre élevé de clients. Or, deux facteurs peuvent limiter la taille de ce marché : **d'une part le type de licence Open Source choisi**, et **d'autre part la présence de concurrents.**

Le choix d'une licence de type copyleftée contaminante limite en effet la taille du marché, dans la mesure où il dissuade une certaine catégorie d'agents de devenir utilisateurs et par conséquent clients potentiels. Ce type de licence exclut du marché, c'est-à-dire du cercle des utilisateurs, les éditeurs de logiciels commerciaux susceptibles d'inclure le logiciel dans leur produit. En effet, son caractère contaminant empêcherait ces derniers d'exploiter commercialement le produit selon un modèle propriétaire traditionnel. Or, ces éditeurs peuvent constituer des clients potentiels, demandeurs de services d'intégration auprès de l'entreprise mettant en place son modèle Open Source. C'est pour cette raison qu'une licence de type libre non copyleftée ou copyleftée persistante peut paraître plus appropriée, pour son caractère non contaminant. C'est d'ailleurs une licence de ce type qui a été choisie pour le produit Open Cascade.

La taille du marché peut également se trouver limitée, à partir du moment où un concurrent apparaît, non pas sur l'activité de fourniture de services, mais sur l'édition d'un logiciel similaire. Les utilisateurs ont alors la possibilité de se tourner vers ce produit concurrent, ce qui peut réduire le marché de l'éditeur, voire même le faire disparaître et par conséquent rendre impossible une quelconque valorisation. Ce risque est d'autant plus grand si l'éditeur se trouve dans une phase de démarrage et le concurrent possède une forte notoriété et/ou un fort pouvoir de marché, lui permettant de concurrencer fortement l'entreprise à l'origine du logiciel open source, grâce à des moyens financiers et marketing supérieurs. Ce risque subsiste quel que soit le type de licence choisi. Si la licence est libre, de quelque type que ce soit, il existe un risque de « clonage » de la part de concurrents : tout logiciel publié en open source par l'éditeur est, par définition, accessible et disponible, y compris à tout concurrent. Ces derniers ont alors la possibilité de réaliser un « forking », c'est-à-dire d'utiliser le code source du produit original et de créer un produit, éventuellement divergent en lui fournissant des évolutions différentes. Le produit devient alors un logiciel concurrent, mais donnant lieu à une valorisation basée obligatoirement sur un modèle open source ; ceci

s'explique par le caractère copylefté de la licence, qui impose de placer tout produit modifié sous licence copyleftée. Si la licence choisie est de type non copyleftée, toute entreprise a la possibilité de s'approprier le produit et, cette fois, de lui appliquer une licence propriétaire permettant une valorisation commerciale traditionnelle.

2.1.3.2. Taux de transformation des utilisateurs en clients

Si la taille du marché constitue une variable à ne pas négliger, ce que l'on peut appeler le taux de transformation des utilisateurs en clients représente un enjeu encore plus important. Un des risques de ce modèle réside dans le fait de confondre utilisateurs et clients. Si les utilisateurs du logiciel open source constituent autant de clients potentiels susceptibles de devenir des clients réels pour le fournisseur de services, ceci n'est pas automatique. Tout l'enjeu pour l'entreprise consiste donc à transformer ses utilisateurs en clients. Si le ratio clients / utilisateurs s'avérait trop faible, la rémunération de l'entreprise serait insuffisante, bien que son logiciel puisse, dans le même temps, être leader sur le marché, voire standard. Or, cette transformation des utilisateurs en clients se heurte à plusieurs difficultés.

La première est liée, à l'environnement concurrentiel de la firme. Cette fois, ce n'est pas sur le produit, mais sur le service que peut exister une concurrence. En effet, le logiciel étant librement utilisable, rien n'empêche alors une société de services concurrente de fournir tout à fait légalement des services identiques basés sur le produit open source. Certes, on peut penser que les compétences techniques détenues par l'éditeur ayant conçu le logiciel pourraient être suffisantes pour inciter les clients à le choisir comme fournisseur de services, mais ce serait oublier l'importance d'autres compétences et atouts tout aussi essentiels : les compétences marketing, la possession d'un fichier de clients, la notoriété. Or, ce type de modèle, comme on l'a vu, implique un changement de métier, d'éditeur de logiciels à fournisseur de services nécessitant d'acquérir en peu de temps ces compétences essentielles. Toute société de services connaissant bien les clients et possédant une forte notoriété est susceptible de capter la valorisation indirecte à son profit, en offrant ses services aux utilisateurs du logiciel open source. Le risque réside donc dans le fait que le bénéficiaire de la valorisation ne soit pas systématiquement l'éditeur ayant investi dans le développement du produit.

La deuxième difficulté susceptible d'empêcher la transformation des utilisateurs en clients est relative aux utilisateurs. En effet, le produit étant open source, rien n'empêche ceux-ci d'adapter eux-mêmes le logiciel à leur activité, sans faire appel à l'éditeur, à partir du moment où ils possèdent les compétences techniques nécessaires en interne. Ils restent alors

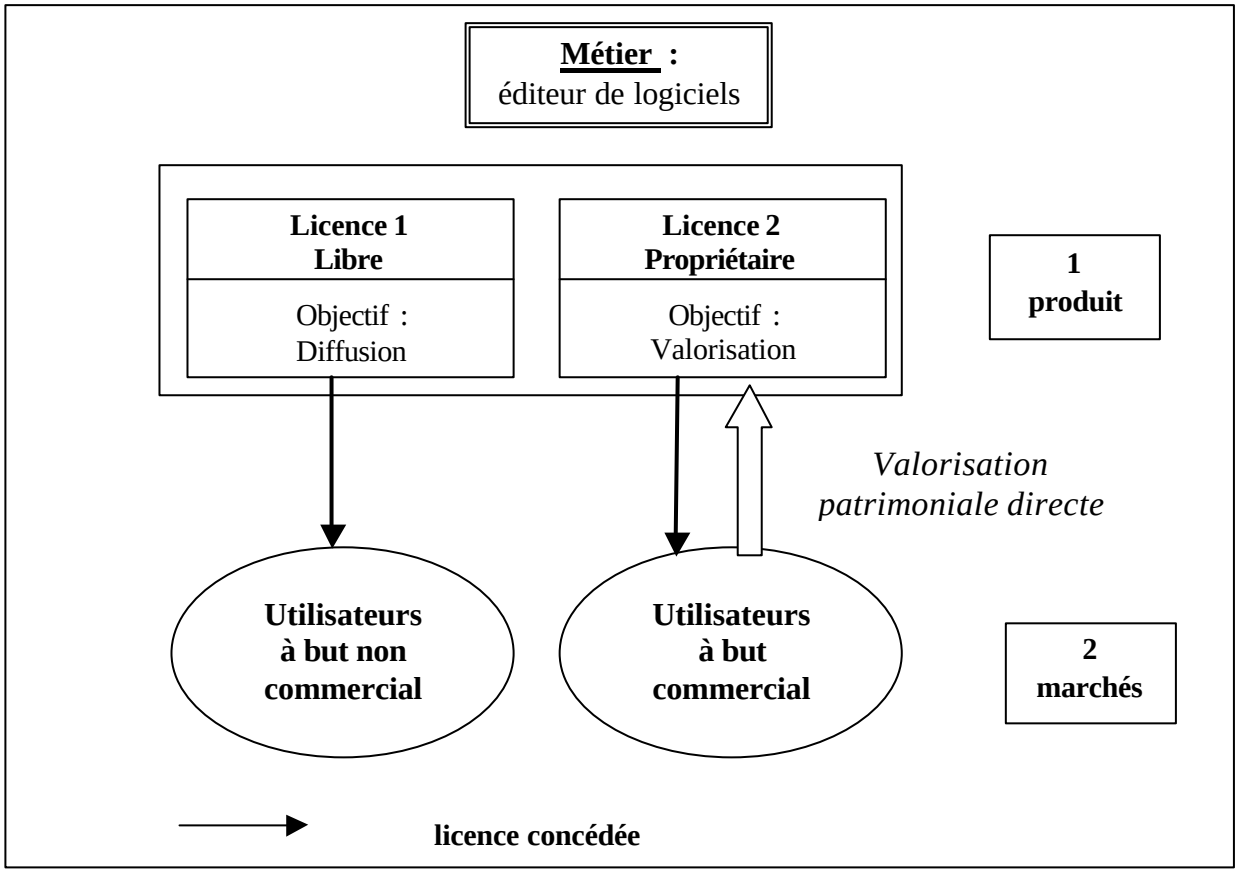
de simples utilisateurs et ne peuvent être comptés parmi les clients de l'éditeur du logiciel / fournisseur de services. Ainsi, lors du choix de ce type de modèle, une réflexion doit être menée sur le degré de « finition » du produit mis en open source, de façon à ne pas fournir en libre un noyau de logiciel trop élaboré. Un logiciel très éloigné d'un produit final sera plus à même de générer de la demande de services de la part des utilisateurs qu'un produit utilisable quasi tel quel, car il demandera des compétences importantes pour être transformé en produit opérationnel. En revanche, un produit quasi final pourra mobiliser des compétences plus simples, détenues en interne par l'entreprise pour être utilisé, ce qui ne permettra pas de transformer ces utilisateurs en clients sources de revenus. Pour créer une demande de services, mieux vaut donc privilégier la mise en open source d'un produit au «degré de finition » peu élevée, soit au niveau technique, soit au niveau informationnel. Au niveau technique, cela peut se traduire par un produit open source fournissant des couches basses (génériques et communes) sans les couches hautes applicatives (spécifiques à l'activité de l'entreprise), dont la mise en place pourrait constituer une demande au fournisseur de services de la part de l'utilisateur. Au niveau informationnel et du savoir-faire, on peut imaginer un produit complexe livré sans documentation ni tutoriaux, pour lequel les utilisateurs pourraient faire appel à la société de services pour une formation.

2.1.4. Types de licences utilisées

Dans une stratégie de valorisation indirecte (par le passage du métier d'éditeur à celui de fournisseur de services), on s'aperçoit que le choix d'une licence libre copyleftée contaminante peut être tout à fait justifié, notamment si le produit est plutôt susceptible d'être utilisé directement par des utilisateurs finals pour les besoins de leur activité. En revanche, à partir du moment où le logiciel constitue plutôt un « outil », module susceptible d'être intégré par d'éventuels éditeurs dans un autre produit destiné à être vendu, le choix de ce type de licence n'est plus judicieux étant donné son caractère contaminant. Une licence libre copyleftée persistante ou non copyleftée peut alors constituer une solution plus adéquate.

Enfin, une licence non copyleftée, si elle convient à ce type de modèle, possède l'inconvénient d'être appropriable et peut ainsi donner lieu à des versions concurrentes propriétaires du produit. Dans le cas de choix de licences copyleftées, persistante ou contaminante, la concurrence n'est pas impossible, mais les versions seront automatiquement libres, imposant un modèle de valorisation open source au concurrent.

2.2. *Valorisation directe d'un produit sur deux marchés discriminés*



Source : Laure Museli

2.2.1. Présentation du modèle

La particularité de ce modèle réside dans le fait qu'un même produit soit proposé à deux catégories d'utilisateurs différentes, à des conditions discriminées. L'entreprise différencie les utilisateurs de son produit et les classe en deux catégories : la première est constituée des utilisateurs à but non commercial, la deuxième des utilisateurs à but commercial. Des conditions d'utilisation discriminées vont alors être appliquées par l'entreprise à ces deux catégories d'utilisateurs, et ceci par l'intermédiaire de deux licences, selon le principe du «dual licensing » (VÄLIMÄKI, 2003). Ainsi, une licence open source, généralement la GPL, est concédée par l'éditeur aux utilisateurs non commerciaux du logiciel, alors que les utilisateurs commerciaux ne peuvent bénéficier que d'une licence propriétaire traditionnelle. La notion d'utilisateurs à but commercial peut varier en fonction des cas. Pour

certains éditeurs, à partir du moment où l'utilisateur vend un produit intégrant son logiciel, il est considéré comme utilisateur à but commercial. Pour d'autres, c'est la fourniture du code-source du logiciel dans le produit final qui constituera le critère d'application de la licence open source ou de la licence propriétaire. Dans ce deuxième cas, si l'utilisateur vend un produit intégrant le logiciel, mais fournit le code-source du module intégré, il pourra bénéficier de la licence open source.

Ce modèle, contrairement à celui que nous avons examiné précédemment, fait appel à un type de valorisation que nous qualifierons de directe. En effet, cette fois, la rémunération de l'entreprise se fait directement par l'intermédiaire de la concession d'une licence propriétaire à certains utilisateurs. L'éditeur garde ici son métier d'origine et ne choisit pas de se diversifier, dans la fourniture de services par exemple, comme dans le modèle précédent.

2.2.2. Intérêt

Le premier intérêt de ce type de modèle est à chercher dans la possibilité de combiner les avantages d'une stratégie open source traditionnelle, permettant coopération avec une communauté de programmeurs et diffusion, et ceux d'une stratégie propriétaire source de valorisation directe par le biais de la licence, sans toutefois changer de métier. Tout d'abord, et comme pour le modèle précédent, le choix d'une licence open source favorise **coopération** d'une communauté de programmeurs **et diffusion**, ce qui peut être intéressant dans une optique de concurrence d'un standard existant ou d'imposition de son produit comme standard. Ensuite, le choix couplé d'une licence propriétaire payante pour les utilisateurs commerciaux donne également la possibilité à l'éditeur d'introduire la composante de **valorisation patrimoniale**, et ceci de façon directe, en lui permettant de générer des revenus grâce à l'exploitation directe de son droit de propriété intellectuelle sur son logiciel. La redevance des utilisateurs à but commercial consiste bien ici en une forme de compensation de la perte de rente de monopole liée au droit d'utilisation cédé au licencié, alors que cette cession de droit reste gratuite pour les utilisateurs à but non commercial, qui n'occasionnent pas de perte de rente de monopole.

Le deuxième intérêt de ce modèle réside dans le fait qu'il ne génère aucune concurrence directe tirant parti des efforts investis dans le produit, à la différence du modèle de valorisation indirecte. Alors que dans ce dernier, la valorisation ne se faisait pas systématiquement au bénéfice de l'éditeur ayant investi dans le développement du produit, mais souvent au profit de concurrents, ceci n'est pas le cas dans ce modèle. Ici, l'éditeur est le seul agent susceptible de récupérer les bénéfices de la valorisation directe de son produit,

puisqu'elle se fait par le biais de la licence, qu'il est le seul à pouvoir concéder. Dans ce modèle, l'exploitation du logiciel dans des solutions commerciales n'est pas néfaste pour l'éditeur, mais encouragée par celui-ci, car source de valorisation par l'intermédiaire de la licence propriétaire payante.

2.2.3. Limites et risques

Si ce modèle de valorisation directe paraît, à juste titre, combiner de nombreux avantages, il comporte également des risques à ne pas négliger.

Le premier déjà évoqué dans le modèle précédent, concerne la concurrence sur le logiciel libre, par le biais du forking. Si ce risque paraît limité dans le cas où l'entreprise exploite depuis longtemps son produit et jouit d'une solide réputation, il prend, comme nous l'avons déjà expliqué, toute son importance lorsque l'entreprise est en phase de démarrage, jeune et méconnue. Il paraît donc essentiel, pour l'éditeur, de se questionner quant au moment opportun de la mise sous licence libre du produit, en fonction de la situation de son entreprise, du type de produit et de l'environnement concurrentiel : suffisamment tôt pour bénéficier des effets positifs induits par l'utilisation d'une licence open source, mais peut-être pas immédiatement, de façon à éviter le clonage par d'éventuels concurrents.

La deuxième limite de ce modèle concerne le type de logiciel produit par l'éditeur. En effet, ce modèle n'est pas applicable en toute circonstance et reste réservé à des éditeurs produisant des composants plutôt que des produits finis. Un «composant » est un produit comportant généralement plutôt des couches basses, non destiné à être utilisé tel quel, mais ayant vocation à être intégré à un logiciel d'utilisation finale. Un composant est donc une sorte d'outil dont d'autres éditeurs, vont se servir pour développer leur propres produits, parfois commerciaux. Un produit fini est, lui, généralement composé de couches basses et hautes qui en font un logiciel d'application, directement utilisable par une entreprise pour les besoins de son activité. Il n'a pas vocation, comme les composants, à être intégré par des éditeurs dans des solutions destinées à être vendues. Pour un composant, le modèle de valorisation directe sur deux marché discriminés est applicable, car si le produit est performant et rencontre une certaine notoriété, des éditeurs l'utiliseront dans leurs logiciels commerciaux, en souscrivant pour cela une licence propriétaire. Dans le cas des produits finis, ce modèle n'est pas applicable, car le marché est uniquement composé d'utilisateurs «non commerciaux », ce qui interdit toute discrimination des conditions d'utilisation et toute possibilité de valorisation patrimoniale du logiciel par le biais de licence propriétaire.

2.2.4. Licence(s) utilisée(s)

Le succès de ce modèle de double licence dépend fortement du type de licence open source choisi. Comme nous l'avons déjà mentionné, il consiste à faire cohabiter deux licences différentes pour un seul produit : une licence open source pour les utilisateurs « non commerciaux » et une licence propriétaire pour les utilisateurs « commerciaux ».

-La licence propriétaire est une licence propriétaire traditionnelle, limitant les libertés d'utilisation, de modification et de distribution du logiciel. La réflexion se situera au niveau du montant et du mode de rémunération adéquats, la licence propriétaire constituant la source de valorisation patrimoniale de ce modèle.

-En ce qui concerne le choix de la licence open source, une licence libre copyleftée contaminante est absolument indispensable au fonctionnement de ce modèle. Elle permet en effet d'imposer la souscription de la licence propriétaire aux utilisateurs commerciaux, car un logiciel sous licence open source contaminantee intégré à un autre programme rend celui-ci obligatoirement open source. Toute entreprise désirant utiliser le module open source tout en gardant son logiciel propriétaire ne pourra utiliser la licence copyleftée contaminante et devra se tourner vers la licence propriétaire. L'utilisation d'une licence copyleftée contaminante garantit donc la composante de valorisation patrimoniale de ce modèle. Imaginons qu'un éditeur adopte ce système de double licence et choisisse une licence copyleftée persistante : l'utilisateur commercial aura alors le choix entre une licence propriétaire payante lui autorisant l'exploitation commerciale d'un logiciel intégrant le composant, et une licence open source autorisant également l'intégration de ce module dans un logiciel et son exploitation commerciale, mais gratuitement cette fois ; il est alors évident que la licence propriétaire ne sera souscrite par aucun utilisateur et la valorisation patrimoniale ne pourra avoir lieu. Le raisonnement reste le même avec l'utilisation d'une licence non copyleftée, non contaminante.

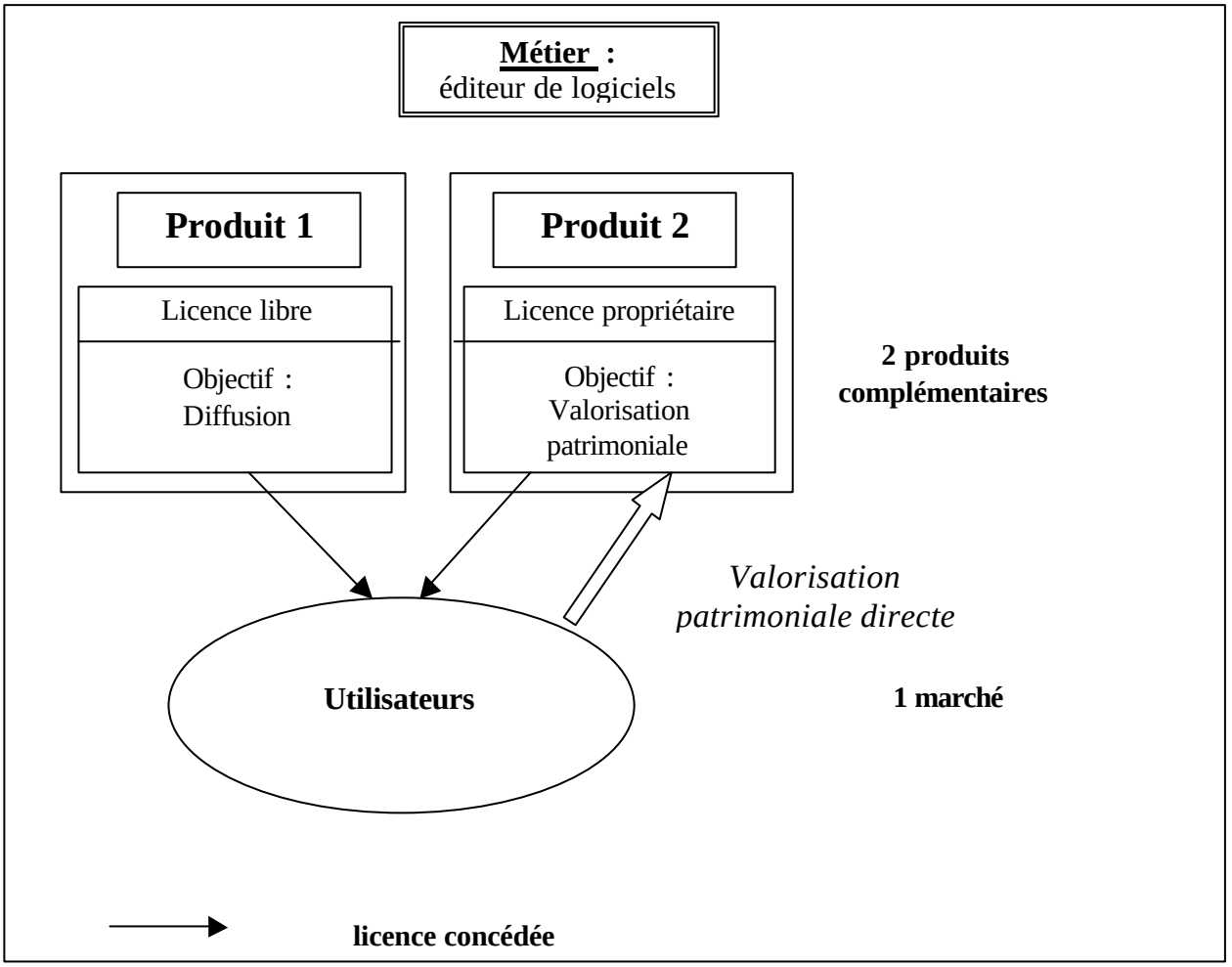
2.2.5. Le cas de Qt de Troll Tech.

Ces licences sont ou ont été appliquées au Qt toolkit, une bibliothèque[vii] utilisée notamment pour créer un Desktop environment[viii], et créée par la société Troll Tech. Cette bibliothèque est destinée à être intégrée dans d'autres logiciels et est notamment utilisée par KDE, un Desktop environment GPL.

Troll Tech a mis au point une politique de licences multiples pour son produit Qt. Elle propose en effet deux licences différentes pour ce même produit. La première est une licence

qu'ils qualifient de libre, appliquée à Qt si les produits qui en sont dérivés ont leur code source disponible : nous l'appellerons désormais la « licence Qt pour logiciels libres ». La deuxième est une licence commerciale que tout utilisateur devra souscrire à partir du moment où il souhaitera distribuer les produits dérivés de Qt sans mettre à disposition leur code source : nous la nommerons la «Qt pour logiciels non-libres ». Ce système a permis à Troll Tech de profiter du pouvoir de diffusion de la licence libre auprès des utilisateurs intégrant Qt dans des produits libres uniquement et d'obtenir une certaine notoriété. L'éditeur profite, dans le même temps, d'une valorisation directe par le biais des redevances tirées des licences propriétaires concédées aux entreprises intégrant sa bibliothèque dans des logiciels commerciaux propriétaires.

2.3. Valorisation directe par vente de produits complémentaires (ou liés) sur un marché unique



Source : Laure Muselli.

2.3.1. Présentation du modèle

Le principe du modèle économique repose ici sur la **combinaison de deux licences différentes pour deux produits complémentaires** au sens large proposés sur **un seul marché**. Le premier produit est proposé par l'entreprise sous licence libre, et satisfait généralement un besoin basique des utilisateurs du marché. Le deuxième produit est placé sous licence propriétaire. Il est complémentaire au sens large, et non au sens strict, car le premier produit peut être utilisé sans le deuxième, mais ce dernier s'avère nécessaire pour des besoins plus évolués de la part des utilisateurs. Les deux logiciels sont donc proposés sur le même marché, mais pour des besoins basiques, un utilisateur aura accès au logiciel open source de façon gratuite, alors que pour des besoins plus évolués, il nécessitera, en plus du

logiciel open source, l'ajout d'une brique supplémentaire payante constituée par le composant propriétaire. C'est ainsi l'utilisation du composant propriétaire qui génère les revenus dans ce modèle de valorisation, alors que le logiciel de base peut être considéré comme un produit d'appel. On peut citer comme exemples de besoin « évolué » la version N+1 d'un logiciel, ou bien une fonctionnalité supplémentaire comme la volonté de jouer en réseau à un jeu vidéo au lieu de jouer seul, ou encore la nécessité de travailler sur un gros site web.

Ce modèle, comme le précédent, fait appel à une valorisation directe, par l'intermédiaire de la concession d'une licence propriétaire aux utilisateurs d'un de ses produits, tout en gardant le métier d'éditeur de logiciels, à la différence du premier modèle.

2.3.2. Intérêt

Ce système de licence permet lui aussi de concilier coopération avec une communauté de programmeurs, diffusion et valorisation patrimoniale, mais cette fois par l'intermédiaire de deux produits d'utilisation liée auprès d'un marché unique rendu captif. Comme le premier modèle, celui-ci se met en place en deux temps. Dans un premier temps, tout l'enjeu consiste, à faire adopter par les utilisateurs le produit de base, afin d'en faire un standard. Le choix de l'open source pour ce produit, outre la coopération qu'il favorise, est expliqué par le pouvoir de diffusion que ce type de licence possède, comme nous l'avons déjà expliqué. La deuxième phase est conditionnée par la réussite de la première : si un nombre important d'utilisateurs adopte le produit de base, qui acquiert des parts de marché importantes, les utilisateurs possédant des besoins plus élevés feront alors appel au produit complémentaire propriétaire. Ici, le couplage de deux produits liés aux licences différentes, l'une gratuite, l'autre payante, est le moyen d'introduire la composante de valorisation patrimoniale dans le modèle utilisant l'open source.

En comparaison avec le modèle précédent, celui-ci n'introduit pas de réelle discrimination entre les utilisateurs en fonction de l'utilisation qu'ils peuvent faire du logiciel. En effet, on ne distingue pas les utilisateurs à but commercial des utilisateurs à but non commercial, ce qui a l'avantage de permettre à ce modèle d'être utilisable par des éditeurs de produits d'utilisation finale, contrairement au système précédent, qui ne pouvait s'appliquer qu'à un éditeur de composants. Le marché cible pour le produit de base est a priori le même que pour le produit complémentaire, et ce ne sont pas les conditions d'utilisation du logiciel qui déterminent son caractère gratuit ou payant. En revanche, c'est le niveau de sophistication du besoin de ces utilisateurs qui détermine la segmentation du marché, en fonction de la nécessité d'utilisation du produit complémentaire ou pas.

2.3.3. Limites et risques

Si ce modèle de valorisation directe par vente de produits complémentaires au sens large sur un marché unique possède les avantages de pouvoir s'appliquer à tout type de produit, final ou composant, et de ne pas imposer de changement de métier à l'éditeur, il ne garantit pas pour autant à celui-ci un succès. Il comporte, comme les autres, certaines limites à ne pas ignorer. Comme dans le cas de la valorisation indirecte par changement de métier d'éditeur à fournisseur de services, l'enjeu principal pour la réussite de l'entreprise utilisant ce modèle concerne le taux de transformation des utilisateurs (clients potentiels) en clients effectifs. De la même façon, l'entreprise devra ainsi s'assurer soit une taille de marché importante, soit un taux de transformation des utilisateurs en clients élevé, sous peine de manquer de clients sources de revenus.

2.3.3.1. Taille du marché

La taille du marché peut se trouver limitée pour deux raisons. La première est relative au choix de licence, pouvant exclure a priori certains utilisateurs du marché. Si le produit de base proposé par l'éditeur est un produit final directement utilisable par le client, ce risque est inexistant. En revanche, s'il s'agit d'un composant, l'utilisation d'une licence copyleftée contaminante exclura de ses utilisateurs les éditeurs à but commercial, pour les mêmes raisons que celles déjà exposées dans le modèle de valorisation indirecte. Le deuxième risque à la limitation de la taille du marché, comme dans le premier modèle, concerne l'apparition d'un concurrent proposant un logiciel de base open source similaire et susceptible de s'imposer sur le marché, aux dépens de l'entreprise. Dans ce cas, la souscription du produit complémentaire ne se fait pas, la valorisation n'est plus possible et l'utilisation de ce modèle un échec.

2.3.3.2. Taux de transformation des utilisateurs en clients

Le taux de transformation peut rester insuffisamment élevé également pour deux raisons.

La première raison concerne le fait qu'un concurrent a la possibilité de décider de profiter du standard open source établi par l'éditeur pour fournir lui-même le composant complémentaire et s'approprier la composante valorisation du modèle. Ce produit complémentaire concurrent peut être de deux sortes : soit propriétaire et issu d'un éditeur commercial, soit libre et issu d'un projet mis en place par la communauté du libre. Dans le cas d'un produit concurrent propriétaire développé par un éditeur commercial, on peut penser que ce dernier aura quelque peu de mal à concurrencer l'éditeur du logiciel open source de base,

qui possède toutes les compétences et la notoriété nécessaires pour que son produit soit choisi par les utilisateurs, au détriment d'un concurrent, pour peu qu'il mette en œuvre des moyens suffisants, notamment en termes de marketing. Le cas de la concurrence du produit complémentaire propriétaire par un produit complémentaire open source mis au point par la communauté du libre n'est pas non plus une hypothèse à négliger. Les promoteurs du Libre, ont en effet tendance à être partisans de solutions «tout open source » et peuvent voir d'un mauvais œil que le produit complémentaire soit propriétaire et profite de l'image libre du produit de base. Il en résulte donc généralement, de la part de telles associations, la mise en place de projets visant à faire développer par la communauté de programmeurs un produit complémentaire concurrent totalement open source. La concurrence d'un logiciel libre peut s'avérer plus dangereuse pour l'éditeur que celle d'un autre logiciel propriétaire, car elle donne la possibilité aux utilisateurs de se procurer le produit à des conditions bien plus intéressantes en termes de prix, et de supprimer ainsi la composante valorisation de l'éditeur.

La deuxième raison pouvant expliquer un taux de transformation insuffisamment élevé tient, une fois de plus, au degré de « finition » du produit. En effet, dans le cas où les fonctionnalités du logiciel de base seraient trop évoluées, elles pourraient satisfaire les besoins d'une majorité des utilisateurs du marché, le produit propriétaire n'étant alors nécessaire qu'à une niche trop petite. Il est donc absolument nécessaire qu'une réflexion soit menée en amont, en ce qui concerne le degré de complétude du produit proposé en libre, afin de créer un marché suffisamment large pour le produit propriétaire. Ceci suppose une analyse précise des besoins des utilisateurs, de façon à délimiter la frontière adéquate entre les besoins satisfaits par le produit libre et ceux nécessitant le produit propriétaire.

2.3.4. Licence(s) utilisée(s)

Comme dans le premier modèle, le choix de la licence open source va dépendre du type de produit proposé. Si c'est un logiciel d'utilisation finale, une licence copyleftée contaminante peut être choisie sans inconvénient. En revanche, s'il s'agit d'un composant, l'entreprise a intérêt à choisir une licence non contaminante de type copyleftée persistante ou non copyleftée.

L'utilisation d'une licence non copyleftée, quant à elle, permet à tout concurrent de s'approprier le logiciel de base et de développer le composant complémentaire permettant de satisfaire des besoins plus évolués. La concurrence, si elle s'exerce au niveau du composant de base, opposera alors un logiciel libre et un logiciel propriétaire possédant les mêmes fonctionnalités, ce qui implique un certain avantage pour le logiciel libre et représente peu

d'intérêt pour le concurrent. On peut alors imaginer que la concurrence se fasse plutôt au niveau du composant complémentaire

2.3.5. Un exemple : Ximian

Ximian utilise pour un de ses produits un modèle de valorisation directe par vente de produits liés sur un marché unique. En effet, il propose Evolution, un logiciel , régi par la GPL, donc une licence libre copyleftée contaminante, et permettant aux utilisateurs de gérer leurs mails et emplois du temps. Evolution concurrence ainsi les produits similaires mais propriétaires de Microsoft : Outlook Mail et PIM (Personal information Manager). Cependant, Evolution est utilisable dans un environnement Linux uniquement. Ainsi, si une entreprise utilise Linux pour l'ensemble de son système informatique, les utilisateurs pourront utiliser Evolution sans aucun problème. En revanche, ce n'est pas le cas si le serveur de mail est celui de Microsoft, MS Exchange.

Ximian propose donc un logiciel passerelle placé sous licence propriétaire permettant de faire cohabiter Evolution avec MS Exchange. C'est la concession de licences d'utilisation payantes de ce module qui constitue la composante de valorisation de ce modèle, alors que Evolution permet notamment de remplir les objectifs de l'entreprise en termes de diffusion.

Le tableau 1 résume quelques unes des conclusions de cet article.

Modèle	Type de licence	Accès marché des éditeurs	Type de logiciel concerné	Produits concurrents dérivés	Concurrents indirects
1	Copyleftée contaminante	Non	Final	Libre	Fournisseur de services
	Copyleftée persistante	Oui	Final ou composant	Libre	Fournisseur de services
	Non copyleftée	Oui	Final ou composant	Libre ou propriétaire	Fournisseur de services
2	Copyleftée contaminante				
	Copyleftée persistante	Oui	Composant	Libre	Aucun
	Non copyleftée				
3	Copyleftée contaminante	Non	Final	Libre	Editeur de logiciels ou projet communauté libre
	Copyleftée persistante	Oui	Final ou composant	Libre	Editeur de logiciels ou projet communauté libre
	Non copyleftée	Oui	Final ou composant	Libre ou propriétaire	Editeur de logiciels ou projet communauté libre

Source : Laure Museli

3. Conclusion

L'apparition de l'open source dans la sphère marchande a donné lieu à un débat récurrent, souvent cantonné à la question « le modèle open source est-il viable ? ».Cet article permet d'avancer quelques éléments de réponse à cette question. Si l'expression « modèle open source » est généralement employée pour qualifier le fait qu'un éditeur commercial utilise, pour l'un de ses produits, une licence open source, tout en parvenant à réaliser des bénéfices, les mécanismes permettant d'atteindre cet objectif commencent à peine à être abordés et restent relativement obscurs. Notre première conclusion consiste à dire qu'il n'existe non pas un, mais plusieurs « modèles open source », basés sur trois modèles génériques, éventuellement combinés par les éditeurs de logiciels commerciaux. Notre deuxième conclusion concerne le fait qu'il soit impossible de répondre de façon manichéenne à la question de la viabilité des modèles open source. Ainsi, on ne peut en aucun cas affirmer que ces modèles open source constituent une panacée et sont adaptés à tous éditeurs de logiciels, tous produits, et viables en toutes circonstances. En revanche, nous nous sommes attachés à illustrer le fait qu'ils puissent, sous certaines conditions, assurer une certaine valorisation, en présentant, pour chacun d'entre eux, les conditions particulières nécessaires à leur succès.

Cet article a tenté également de souligner les implications stratégiques que peut comporter, pour un éditeur de logiciels, le choix de la licence informatique. Il convient en effet, pour l'entreprise, de sélectionner très minutieusement le type de licence à appliquer à son logiciel, en fonction de différents paramètres tels que la nature du produit, la nature et la taille du segment de clientèle à cibler, ou encore l'environnement concurrentiel. Loin d'être anodin, le choix d'une licence open source obéit donc à des motivations stratégiques précises. Si nous les avons seulement évoquées dans cet article, à travers les allusions aux volontés de diffusion et de coopération avec une communauté de programmeurs, cette question mérite d'être développée plus encore[ix].

Enfin, l'analyse que nous proposons reste, à dessein, centrée sur les produits, de par notre volonté d'insister sur la relation entre modèles économiques open source et licences. Nous avons ainsi pris le parti de laisser de côté l'aspect « clients », sans toutefois nier le fait qu'il s'avère important dans la problématique de l'open source. En effet, dans un domaine où les choix d'investissement et les décisions des Directions des Systèmes d'Information sont orientés par un besoin de garantie et de sécurité, la question de la perception, positive ou négative, de l'open source par ce type d'acteur, si nous ne l'avons pas abordée ici, reste cruciale.

Bibliographie

BONACCORSI A. et ROSSI C.,(2003), « Licensing schemes in the production and distribution of Open Source software. An empirical investigation ». http://opensource.mit.edu/papers/bonaccorsirossilicense.pdf, visité le 27 nov. 2003.

CORIS M. (2002), «Les sociétés de service en logiciels libres : l'émergence d'un système de production alternatif au sein de l'industrie du logiciel ? ». ». In Jullien & al. [2002], éditeurs, *Nouveaux modèles économiques, nouvelle économie du logiciel*, pp. 87- 104.

DALLE J.M. et DAVID P., (2003), « The allocation of software development resources in « Open Source » production mode », MIT Working Paper, http://opensource.mit.edu/papers/dalledavid.pdf, visité le 27 nov 2003.

DALLE J.M. et JULLIEN N., (2003), « Libre software: turning fads into institutions ? », Research Policy 32: 1-11.

FREE SOFTWARE FOUNDATION, « Catégories de logiciels libres et non libres », http://www.gnu.org/philosophy/categories.fr.html, visité le 27 nov. 2003.

GARUD et KUMARASWAMY, 1995 ; « Technological and Organizational designs for Realizing Economics of Substitution », Strategic Management Journal, vol. 16, Summer Special Issue, pp. 93-109.

HAWKINS R., (2003), «The economics of free software for a competitive firm », MIT Working Papers, http://opensource.mit.edu/papers/hawkins.pdf, visité le 27 nov. 2003.

JULLIEN N., (2003), "Le marché francophone du logiciel libre", Système d'Information et Management, n°1, vol. 8, pp 77-100

LEE S.H., (1999), « Open Source Software licensing », http://eon.law.harvard.edu/openlaw/gpl.pdf, visité le 27 nov. 2003.

LERNER J., TIROLLE J., (2002 a), « The scope of Open Source Licenses », MIT Working Paper, http://opensource.mit.edu/papers/lernertirole2.pdf, visité le 27 nov 2003.

LERNER J., TIROLLE J., (2002 b), « The Simple Economics of Open Source », *Journal of Industrial Economics*, 52, pp.197-234.

MUSELLI L., (2002), « La licence: un outil stratégique pour les éditeurs de logiciels ». In Jullien & al. [2002], éditeurs, *Nouveaux modèles économiques, nouvelle économie du logiciel*, pp. 129-145.

MOCKUS A., FIELDING R. et HERBSLEB J., 2000 ; « A case study of open source software development : the Apache Server », Proceedings of the 22nd internation conference on Software engineering, pp. 263-272.

OSI (OPEN SOURCE INITIATIVE), http://www.opensource.org/, visité le 27 nov. 2003.

PAL et MADANMOHAN, (2002), «Competing on Open Source : Strategies and Practise », MIT Working Paper, http://opensource.mit.edu/papers/madanmohan.pdf, visité le 27 nov. 2003.

PERENS B., (1998), «Définition de l'open source – version 1.0 », traduit en français par S. Blondeel, http://www.linux-France.org/article/these/osd/fr-osd.html, visité le 05 avril 2004.

PRASAD G., (2001) "Open Source-onomics: Examining some pseudo-arguments about Open Source, Linux Today" , http://linuxtoday.com/news_story.php3?ltsn=2001-04-12-006-20-OP-BZ-CY, visité le 5 avril 2004.

RAYMOND E.S., (1999), « The Magic Cauldron », http://www.catb.org/~esr/writings/magic-cauldron/, visité le 05 avril 2004.

RAYMOND E.S., (2001), « The Cathedral and the Bazar. Musings on Linux and Open Source by an accidental revolutionary », O' Reilly Sebastopol, CA.

ROGERS E. M., (1995), "Diffusion of Innovations", 4th Edition, New York :Free Press.

STEINMULLER E., GARCIA J., (2003), « The Open Source Way of Working : A New Paradigm for the Division of Labour in Software Development ? », SPRU Electronic Working Paper n°92.

TROLLTECH, « Licenses for code used in Qt », http://doc.trolltech.com/3.1/licenses.html, visité le 27 nov. 2003

VON HIPPEL E., (2001), « Innovation by user communities : Learning from open source software », Sloan Management Review, 42 (4), pp.82-86.

SHAPIRO C., VARIAN H., (1999), *Economie de l'information, Guide stratégique de l'économie des réseaux*, De Boeck universités, Bruxelles. Traduction française de Shapiro et Varian [1999].

VÄLIMÄKI M. (2003) «Dual Licensing in Open Source Software industry », Systèmes d'Information et Management, 1/2003

VAN WENDEL DE JOODE, R., DE BRUIJN J.A., VAN EETEN M.J.G., (2003) « Protecting the Virtual Commons, Self-organizing open source and free software communities and innovative intellectual property regimes », Information Technology and Law Series, number 3, T.M.C. Asser Press, The Hague.

http://www.opencascade.com, visité le 27 nov. 2003.

http://www.ximian.com, visité le 27 nov. 2003.

Notes :

[i] L'absence d'accès au code source est ainsi à rapprocher du « secret ». Elle a pour but de renforcer la protection fournie par le droit d'auteur, que les éditeurs de logiciels propriétaires trouvent trop faible. En effet, le droit d'auteur protège l'expression du code, mais pas les fonctionnalités du programme, ce qui permet à tout programmeur ayant accès à un code source, de l'étudier et de réécrire un logiciel aux fonctionnalités équivalentes, sans pour autant enfreindre le droit d'auteur.

[ii] Notre article se concentre sur les modèles open source adoptés par les éditeurs de logiciels commerciaux uniquement, c'est-à-dire soucieux de valoriser des logiciel open source développés par leurs soins, et élimine volontairement du champ de l'étude les entreprises susceptibles de tirer des revenus de logiciels open source développés par d'autres éditeurs. Par exemple, les cas des entreprises telles que Red Hat ou Mandrake, ne seront pas abordés ici, car ces sociétés ne sont pas éditrices du logiciel Linux, mais simplement distributrices d'une version packagée de ce dernier.

[iii] En Europe, des débats sur la brevetabilité des logiciels sont encore en cours et la législation retient toujours le droit d'auteur comme unique protection intellectuelle des logiciels en tant que tels . Pour cette raison, nous nous en tiendrons au seul droit d'auteur comme droit de propriété intellectuelle appliqué au logiciel dans cet article.

[iv] Pour de plus amples détails concernant la définition de l'Open Source et du Libre, on pourra se référer à Perens (1999), ainsi qu'aux sites de l'OSI et de la FSF.

[v] Pour une étude de l'activité de ces SSLL, on pourra se référer notamment à CORIS (2002).

[vi] Ce type d'utilisateur correspond à la catégorie que Varet et Zimmermann (1985) nomment « utilisateurs designers ». Ceux-ci sont *capables de travailler directement sur les technologies d'architecture et sur les technologies d'utilisation*. Ce sont, par exemple, *les ingénieurs réseau qui vont co-construire la partie physique du réseau informatique et participer au choix (et à l'adaptation) des logiciels qui vont rendre les «services» demandés.*

[vii] Une bibliothèque logicielle est un ensemble de code assemblé pour réaliser un groupe de tâches liées. Les bibliothèques logicielles se distinguent des exécutables (fichiers contenant des commandes entraînant des actions ou des opérations de la part du système) dans la mesure où elles ne sont pas des programmes indépendants ; elles sont plutôt du code "assistant" un programme indépendant en lui fournissant des services. [Wikipedia, L'Encyclopédie du Libre]

[viii] Un desktop environment sert à configurer l'aspect graphique d'un écran et est composé d'un Window manager (permettant de configurer l'encadré et le gestionnaire de fenêtre) ainsi que d'outils comme des barres ou des menus déroulants améliorant la convivialité de l'environnement.

[ix] Cette question de l'implication stratégique des choix de licence par les éditeurs de logiciels fait d'ailleurs l'objet de Muselli (2002).