N° attribué par la bibliothèque

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Ecole Doctorale n°396 (Economie, Organisations & Société)

THESE

pour obtenir le grade de

Docteur de Mines ParisTech

Spécialité « Sciences de Gestion »

présentée et soutenue publiquement par

Nordine BENKELTOUM

Le mercredi 9 décembre 2009.

LES REGIMES DE L'OPEN SOURCE : SOLIDARITE, INNOVATION ET MODELES D'AFFAIRES.

Directeur de thèse : Armand HATCHUEL

Jury :

Frank AGGERI		Professeur à Mines ParisTech.
Albert DAVID	Rapporteur	Professeur à l'Université Paris-Dauphine.
Armand HATCHUEL		Professeur à Mines ParisTech.
Grégory LOPEZ		Responsable du Centre de Compétences Open source de Thales D3S.
François Xavier de VAUJANY	Rapporteur	Professeur à l'Université Paris-Dauphine.

AVERTISSEMENT

Mines ParisTech n’entend donner aucune approbation ni improbation aux opinions émises dans cette thèse. Ces opinions doivent être considérées comme propres à l’auteur.

REMERCIEMENTS

Du point de vue académique, je tiens tout d'abord à exprimer ma reconnaissance à Armand Hatchuel qui a réellement donné le temps et l'attention nécessaires au bon déroulement de mon travail de thèse. Je suis tout particulièrement reconnaissant pour le rôle structurant de ses orientations méthodologiques et bibliographiques.

J'adresse également mes chaleureux remerciements à Pascal Roos sans qui ce travail n'aurait pu voir le jour ainsi qu'à Christian Simon pour son soutien constant et inchangé au cours de ces dernières années.

J'exprime ma gratitude à Jean-Claude Sardas et Olivier Favereau pour le soutien qu'ils ont apporté à mon projet de recherche.

Je remercie Akin Kazakci et Blanche Segrestin pour les discussions informelles que j'ai eu le plaisir de partager avec eux.

Merci également à Mathias Béjean, Sébastien Gand, Mathias Szpirglas pour leur aide, leurs pertinentes remarques au cours de mes interventions. Je remercie aussi les doctorants du CGS pour nos échanges et plus particulièrement Emmanuel Coblence, Sophie Hooge, Paris Chrysos et Cédric Dalmasso.

Je tiens aussi à formuler un grand merci à l'équipe administrative du CGS : Martine Jouanon, Céline Bourdon et Christine Martinent pour leur efficacité et leur professionnalisme.

Je remercie également Pascal Le Masson et Benoit Weil du CGS, Dominique Frugier et Michel Bigand de l'Ecole Centrale de Lille pour m'avoir donné la possibilité d'enseigner mes recherches sur la stratégie d'entreprise et l'innovation.

J'exprime aussi mes profonds remerciements à Benjamin Grassineau pour sa précieuse aide pour la relecture des premières versions de ce manuscrit et pour nos discussions *« énergiques »* sur l'open source. Merci aussi à Rémi Bachelet pour son accueil à l'Ecole Centrale de Lille et nos différentes collaborations. J'adresse aussi une pensée à Vincent Meissner du secrétariat du Laboratoire de Génie Industriel de Lille pour son aide et son grand professionnalisme.

Je transmets également mes remerciements à Albert David et Sébastien Damart ainsi qu'aux participants du CDEG (Cercle Doctoral Européen de Gestion) et tout particulièrement à Loïc Cadin, Gérard Charreaux, Gilles Garel et Martine Girod Seville pour leurs pertinentes remarques et suggestions lors de notre présentation à mi-parcours (Benkeltoum 2008a).

Je remercie de même John Christiansen et les participants de l'atelier doctoral de la conférence IPDM (International Product Development Management) qui fut très riche en débats et en enseignements (Benkeltoum 2008b).

Je tiens à remercier Donald White pour son aide en écriture scientifique anglo-saxonne et ses suggestions, Paul Duguid pour ses explications sur les communautés de pratique et Frank Piller pour ses intéressants commentaires sur nos différents travaux.

Du point de vue empirique, cette thèse n'aurait pas été possible sans le soutien des professionnels de l'open source.

J'exprime tout d'abord mes remerciements à Grégory Lopez, Responsable du Centre de Compétences Open Source au sein de Thales D3S pour le soutien institutionnel de Thales D3S, son implication dans la mise à disposition de son réseau d'industriels, ainsi que pour sa disponibilité malgré ses responsabilités.

Je remercie ensuite Serge Druais, Directeur de la Recherche et de l'Innovation de Thales D3S, pour le temps qu'il a bien voulu consacrer à notre recherche et pour ses contributions pertinentes et éclairantes à notre thèse.

Je remercie également Viet Ha et Julie Marguerite, Open Source Architects de Thales D3S, pour le temps qu'ils ont bien voulu m'accorder pendant mes recherches empiriques.

Je suis particulièrement reconnaissant envers Laurent Clévy, Chercheur Industriel de Bell Labs pour sa précieuse expertise de l'industrie informatique et de l'open source en particulier.

Merci également à Marc-Aurèle Darche, Membre du conseil d'administration de l'AFUL pour nos différents échanges et discussions animées sur l'open source. Sans oublier également Sophie Gautier, Membre du community council d'OpenOffice.org pour le temps qu'elle a bien voulu accorder tout au long de notre recherche.

Je remercie de même Eric-Marc Mahé (Sun Microsystems), Jean-Noël De Galzain (Wallix), David Ascher (Mozilla Messaging Inc.), Tristan Nitot (Mozilla Europe) Jean-Michel Delettre (Freeworks) François Bancilhon (ex-Mandriva), Pierre-Yves Gibello (Experlog) et Serge Lacourte (ScalAgent Distributed Technologies) pour le temps qu'ils ont bien voulu accorder pour m'expliquer le fonctionnement de leur organisation respective.

J'exprime une profonde gratitude à l'ensemble des contributeurs à cette thèse dont les apports trouvent un remerciement symbolique à la fin de ce document où nous avons inscrit les noms reproduisant ainsi l'esprit ouvert de l'open source.

Bien que ce travail ait reçu la contribution de différentes personnes, j'assume l'entière responsabilité de toutes les erreurs que ce document pourrait contenir.

Nordine Benkeltoum.

LES REGIMES DE L'OPEN SOURCE : SOLIDARITE, MANAGEMENT ET MODELES D'AFFAIRES.

A l'origine, l'industrie informatique confondait le matériel (*« hardware[1] »*) et le logiciel (*« software »*) en une seule et même économie (Dalle, David, Ghosh, et Steinmueller 2005: 413). Les fabricants n'avaient aucun intérêt à vendre composants et logiciels séparément puisqu'ils n'étaient pas utilisables individuellement du fait de l'hétérogénéité des systèmes et de la non compatibilité des instructions écrites entre ces derniers. C'est pourquoi, il n'y avait pas vraiment d'économie du logiciel autonome (Rannou et Ronai 2003b: 20). Par exemple, la Star Workstation de Xerox intégrait aussi bien du matériel que du logiciel (Chesbrough 2003: 78).

L'une des conséquences de la *« non valorisation »* autonome du logiciel sur le marché, se traduisait par un phénomène de partage de code entre les utilisateurs. En effet, pendant cette période les utilisateurs étaient principalement des professionnels ou des chercheurs académiques ayant des connaissances pointues en informatique. Le partage de code source n'est donc pas un phénomène récent. Selon Richard Stallman, il s'agissait d'une pratique courante dans les années 70 (Nuvolari 2003: 5). De même en 1967, Robert Fano du MIT expliquait que les utilisateurs bâtissaient des logiciels en se basant sur le travail des uns et des autres. Il ajoutait que plus de la moitié des commandes du système d'exploitation à temps partagé, utilisé au MIT, furent développées par des utilisateurs (Ilkka Tuomi 2005: 430). L'essentiel des échanges était réalisé sur support matériel puisqu'il n'y avait pas encore d'infrastructure réseau permettant l'échange en ligne : ceci correspond à la première phase de développement du phénomène open source (*Phase 1 : l'échange matériel de code informatique*).

Parallèlement à cette situation, plusieurs changements technologiques eurent des répercussions sur la structure de l'industrie informatique. Le premier changement connu par cette industrie fut, sans commune mesure, l'évolution des langages de programmation provoquant une séparation progressive entre les instructions écrites en *« langage homme[2] »* et les instructions exécutées sur la machine alors qu'auparavant le premier langage de programmation « *se rapprochait plus de la machine*[3] ». Le point de départ de cette séparation est la création du premier compilateur[4] par Grace Hopper en 1951.

Le second changement majeur apparu dans l'industrie informatique est l'émergence de standards de fait (Osterloh et Rota 2007: 163). Pour ne citer que les plus centraux : le PC

[1] Hardware désigne du matériel informatique.
[2] Le langage homme correspond aux instructions écrites dans un langage de programmation compréhensible par l'homme.
[3] Nader, TDR, entretien téléphonique avec l'auteur, mercredi 29 novembre 2006.
[4] Un compilateur est un programme permettant de convertir des instructions écrites dans un langage de programmation en langage machine ou binaire.

(Personal Computer), Unix et Windows. Le PC (inventé par IBM) est devenu une infrastructure standardisée pour la quasi-totalité de l'industrie. De ce fait, les acteurs de l'industrie informatique devaient concevoir leurs composants en respectant certaines normes de conception. Le système d'exploitation Unix (commercialisé par Bell Labs) était distribué sous une licence permettant à la fois la lecture, la modification et le partage du code source. Ce système fut donc décliné en de nombreuses versions plus ou moins dérivées : BSD, Solaris, Xenix, Mac OS X, Minix, Linux, etc. Windows, à travers ses différentes versions, eut un succès sans précédent dans l'histoire des systèmes d'exploitation. Suite à ces changements technologiques, l'industrie informatique s'est scindée en deux parties : l'industrie des composants et celle des logiciels.

C'est seulement à partir du moment où les concepteurs de logiciels eurent la possibilité de convertir les instructions lisibles par l'homme (*« code source*[5] *»*) en instruction machine (*« code objet*[6] *»*) par le biais du compilateur, que ces derniers eurent la possibilité de *cacher* la source de leur travail à leurs clients et plus largement à tous les utilisateurs. Du fait de ces changements, une nouvelle dichotomie s'opéra au sein même des logiciels : les logiciels dont le code source était disponible et modifiable ; les logiciels dont le code source était ni disponible, ni modifiable. En fait, ce qui fut inventé ce n'est pas le logiciel libre, c'est plutôt le logiciel fermé puisqu'au commencement de l'informatique les logiciels étaient libres. Actuellement, la séparation entre le langage de programmation et langage machine est de plus en plus marquée grâce aux progrès scientifiques réalisés dans ces cinquante dernières années (Bancilhon 2007).

Par conséquent, l'industrie du logiciel s'est elle-même séparée en deux parties appelées plus tardivement : logiciels libres[7] et logiciels fermés. Selon la Free Software Foundation[8] (FSF), un logiciel est considéré comme libre s'il possède quatre libertés : l'exécution, l'étude, l'amélioration et la distribution. A contrario, un logiciel non libre se définit comme un logiciel auquel il manque une ou plusieurs de ces *libertés*. Dans la majorité des cas, les licences des

[5] Un code source est un ensemble d'instructions écrites dans un langage de programmation.
[6] Un code objet est le résultat d'un processus de compilation de code source. Il représente des instructions binaires c'est-à-dire une suite de caractères composée de « 0 » et « 1 ».
[7] Pour une introduction générale sur les logiciels libres nous invitons lecteur intéressé à lire (Guérin 2001; Rastetter 2002).
[8] La Free Software Foundation est une organisation à but non lucratif fondée par Richard Stallman. Elle assure la promotion et la défense des logiciels libres au niveau international. Elle est à l'origine de la General Public Licence (GPL), la licence la plus populaire dans le domaine du logiciel libre. La GPL est un instrument juridique visant à protéger les quatre libertés du logiciel libre.

logiciels non libres[9] donnent uniquement le droit à une exécution limitée[10] en restreignant les droits des utilisateurs en termes de modification du programme (Johnson, 2002 : 638).

En 1969, la création d'Arpanet permettait aux chercheurs universitaires de partager du code mais seulement pour des fragments car Arparnet avait des capacités limitées en termes de bande passante. A ce moment les firmes étaient peu impliquées dans les échanges et concernaient principalement des universitaires ayant accès à ce réseau. Pour preuve, le projet GNU[11] est issu du milieu universitaire. Ce projet fût initié en 1983 par R. Stallman[12] alors qu'il faisait parti d'un laboratoire d'intelligence artificielle du MIT, celui-ci disait : *« commençant ce Thanksgiving je vais écrire un système complet compatible avec Unix appelé GNU (pour Gnu's Not Unix), et donner celui-ci gratuitement à tous ceux qui veulent l'utiliser »* (Stallman 1983). Il s'agit de la seconde phase de développement de l'open source (*Phase 2 : l'échange partiellement électronique de code informatique*).

Jusqu'au début des années 90, les logiciels libres n'étaient pas intégrés dans une dynamique économique. L'émergence du système d'exploitation *« GNU/Linux »* né de la convergence entre le projet GNU et le noyau Linux initié par Linus Torvalds[13], bouleversa une bonne partie des composantes de l'industrie informatique.

Grâce à la propagation d'internet, le phénomène open source n'était plus seulement cantonné au milieu universitaire puisque n'importe quel développeur pouvait télécharger du code et procéder à des modifications ou encore partager du code qu'il avait initialement réalisé pour ses propres usages. Cette phase de démocratisation s'est largement fondée sur le développement d'internet à la fois le fruit et l'engrais du développement de l'open source (Deek et McHugh 2007: 11). Le TCP/IP[14], le HTML[15] et BIND[16] sont quelques-uns des

[9] Les logiciels non libres sont aussi appelés logiciels closed source (Johnson, 2002 : 638). Nous utiliserons ces expressions de manière interchangeable dans notre travail. A noter également que l'on rencontre fréquemment la notion de *« logiciel propriétaire »* qui porte selon nous à confusion car le logiciel libre a aussi un ou des propriétaire(s). C'est la licence qui donne au logiciel sont statut libre ou non libre et non pas le fait qu'il soit détenu par un propriétaire.

[10] L'exécution peut être limitée par exemple en nombre de postes ou d'utilisateurs.

[11] Gnu is Not Unix ou GNU est un projet initialement fondé dans le but de créer un système d'exploitation libre. Historiquement, le Projet GNU connaissait des difficultés pour concevoir *Hurd* : le noyau du système. Linux, un noyau initié par Linus Torvald, a remplacé le balbutiant *Hurd* donnant naissance au Projet GNU/Linux : le premier système d'exploitation entièrement libre.

[12] Richard Stallman est une figure emblématique à l'origine de la notion de « *Free Software* » et du Projet GNU. Il est connu dans le monde de l'informatique pour ses capacités techniques. Aujourd'hui, il fait toujours preuve d'un activisme zélé pour la promotion et la défense des logiciels libres.

[13] Linus Torvald est l'initiateur du noyau Linux devenu célèbre pour ses capacités techniques d'intégration de contributions.

[14] Le TCP/IP (Transmission Control Protocol/Internet Protocol) est un protocole de communication conçu en 1974 afin de contourner la disparité des systèmes.

[15] HTML (Hypertext Markup Language) est un langage de balisage permettant d'afficher des pages web.

[16] BIND (Berkeley Internet Name Domain) est un système d'adressage des sites internet permettant de ne pas avoir à entrer l'adresse au format numérique mais sous forme de noms de domaines.

standards libres sur lesquels internet est basé illustrant bien le fait que le *« réseau des réseaux[17] »* est le produit du phénomène open source (*Phase 3 : la phase de démocratisation*).

En 1998, un événement important marqua l'entrée des firmes dans l'open source : la libération de Navigator par Netscape. La firme avait en effet libéré son navigateur internet en réaction à la stratégie de Microsoft qui embarquait gratuitement Internet Explorer dans son système d'exploitation. D'après L. Torvald, la libération de Navigator constituait une stratégie de la dernière chance (mentionné par Torres-Blay (2003: 16)). Le second événement important de la fin des années 90 est la création de SourceForge.net en 1999. SourceForge.net proposait gratuitement aux programmeurs de logiciels libres un ensemble d'outils nécessaires au développement et à la diffusion de leurs créations.

A cette même période, la bulle spéculative liée aux NTIC (Nouvelles Technologies de l'Information et de la Communication) avait de sérieuses répercussions sur les *« dot.com[18] »* provoquant des faillites en chaîne. En revanche, les organisations de l'open source ne furent pas vraiment touchées par l'éclatement de cette bulle car pour la plupart, elles n'avaient pas de modèle d'affaires et étaient encore moins cotées sur les marchés financiers. Au fur et à mesure les firmes commencèrent à libérer des logiciels fermés et à participer aux communautés existantes. C'est ce qui correspond à la quatrième phase du développement de l'open source (*Phase 4 : la phase de coopération*).

Pendant la phase de coopération, les firmes ont appris comment gérer des communautés distantes et ont par conséquent copié le modèle des communautés d'utilisateurs-développeurs pour leurs propres développements. Ainsi, les firmes commencèrent à rationnaliser le développement de logiciels libres. Ceci correspond à la cinquième phase de développement de l'open source (*Phase 5 : la phase d'industrialisation*). Les entreprises sont passées d'observateurs passifs, au statut d'investisseurs et désormais au statut de principaux acteurs. Pour reprendre la terminologie de Lerner et Tirole, les entreprises sont passées d'une *« reactive strategy »* à une *« proactive strategy »* (Lerner et Tirole 2000: 26-27).

Depuis que Linus Torvald mit à disposition du public le noyau de système d'exploitation sur lequel il travaillait, le phénomène a connu de profondes mutations. Selon nos analyses, ces modifications importantes sont principalement dues à l'implication croissante de firmes dans le phénomène et à l'évolution technologique. Le tableau ci-dessous résume l'évolution du

[17] L'expression *« réseaux des réseaux »* est l'autre appellation d'internet.
[18] Les *« dot.com »* sont les start-up ayant émergé lors de la phase d'engouement liée au développement d'internet.

phénomène open source en mettant en évidence l'impact croissant des firmes dans cette industrie.

Tableau 1 : L'évolution de l'open source.

	Phases de développement	Acteurs principaux	Evénements importants	Période	Rôle des entreprises
Phase 1	Phase d'échange matériel de code informatique.	Chercheurs universitaires	/	Avant 1970	Faible.
Phase 2	Phase d'échange électronique de code informatique (partielle).	Chercheurs universitaires	Arpanet (69) ; TCP/IP (72) ; CNET (79).	1970 - 1990	Faible.
Phase 3	Phase de démocratisation	Utilisateurs-développeurs	Popularisation d'internet.	1990 - 1998	Premiers investissements dans le phénomène
Phase 4	Phase de coopération	Utilisateurs-développeurs et entreprises	Libération de Netscape Communicator (1998) ; Création de SourceForge.net (1999) Eclatement de la bulle internet (2000).	1998 - 2002	Création d'organisations avec des acteurs individuels.
Phase 5	Phase d'industrialisation	Entreprises	Création d'ObjectWeb (2002)	Depuis 2002	Libération massive de code ; création de communautés inter-organisations et de consortiums.

Depuis le début de ce XXI^{ème} siècle, le logiciel libre est devenu un élément incontournable de l'industrie du logiciel. Des firmes ont rapidement bâti des modèles économiques[19] adaptés aux propriétés des logiciels libres. Différentes études ont chiffré à plusieurs milliards leur potentiel économique : l'IDC estime qu'en 2011 les revenus liés aux logiciels libres devraient atteindre 5,8 milliards de Dollars (IDC 2007).

Les logiciels libres (traduit de *« Free Software »*) sont aussi appelés logiciels *« Open Source »*. La notion de logiciel *« Open Source »* fut créée par Bruce Perens car l'expression *« Free Software »* faisait apparaître en anglais une ambigüité entre liberté et gratuité. La volonté des partisans de l'open source était de promouvoir les logiciels libres auprès des entreprises en créant l'Open Source Initiative (OSI)[20].

[19] La notion de *modèle économique* est une traduction de l'expression « *business model* ». Selon Chesbrough, un business model a deux fonctions : créer et capturer de la valeur (Chesbrough, 2007 : 12).

[20] L'OSI est une organisation à but non lucratif créé afin de promouvoir les logiciels libres dans le domaine professionnel. La principale différence entre l'OSI et la FSF concerne leur vision respective des logiciels libres

D'après Loïc Dachary, Fondateur de la FSF Europe, *« le logiciel libre dit que la liberté du logiciel est une affaire de philosophie et l'open source s'en distingue car il rejette toute philosophie*[21]*. C'est le motif de création du mouvement open source, de rejeter la philosophie du logiciel libre. Les auteurs, fondateurs du mouvement open source ne s'en sont pas cachés du tout*[22]*. »* D'après la FSF, aujourd'hui le mouvement de l'*Open Source* et celui du *Free Software* sont séparés (Free Software Foundation 2005: 54). Toutefois selon d'autres *« les définitions de l'OSI […] et de la FSF sont les mêmes, et pour moi, le distinguo fait entre les deux est artificiel, et purement sur la forme*[23]*. »* Pour ne pas rentrer dans ce débat terminologique, nous considérerons que *« Free Software », « Open Source Software »* et *« Logiciel Libre »* sont synonymes. De ce fait, nous utiliserons de manière interchangeable ces différentes expressions, conformément à l'avis de la plupart des universitaires de ce champ de recherche (von Hippel et von Krogh 2003: 210).

L'analyse de l'impact des logiciels open source est un objet d'étude complexe puisque celui-ci met en relation des domaines hétérogènes. Tout d'abord, le logiciel libre met en jeu le domaine technologique puisqu'un logiciel est avant toute chose un objet virtuel. Ensuite, le logiciel libre défie l'économie et les modèles d'affaires étant donné qu'aujourd'hui *« l'open source ce n'est plus seulement des particuliers, les entreprises se sont appropriées le phénomène pour faire de l'argent*[24] ». Puis, il interroge le droit parce que les licences des logiciels libres ont ouvert de nouveaux débats juridiques[25]. Enfin, il pose de nouvelles questions en sciences de l'organisation car la production du logiciel libre a provoqué une remise en question des formes productives existantes en couplant à la fois les propriétés d'un mouvement social (O'Mahony 2002), d'un réseau coopératif (Grassineau 2009), d'un régime de solidarité ouvert et d'une structure productive distribuée. Afin de rendre compte de cette richesse dans notre travail, nous avons fait le choix d'avoir des méthodes et des sources pluridisciplinaires.

Dans cette thèse, nous détaillerons ce que nous nommons *les régimes de l'open source.* Mais qu'entendons-nous par cette expression ? Par *régimes de l'open source,* nous faisons

---

et fermés : pour l'OSI, un logiciel fermé est simplement un choix d'auteur et une méthode de développement ; tandis que pour la FSF, le logiciel fermé est un problème social.

[21] La philosophie du logiciel du libre repose sur les quatre libertés du logiciel.

[22] Loïc Dachary (Fondateur et vice-président, FSF Europe), entretien en face à face avec l'auteur, mardi 21 novembre 2006.

[23] Cooker, Communauté Mandriva, Entretien en ligne avec l'auteur, Lundi 15 mai 2006.

[24] Grégory Lopez, Responsable du Centre de Compétences Open Source, Thales D3S, entretien en face à face avec l'auteur, mardi 23 janvier 2007.

[25] A ce sujet, voir le travail pionnier de M. Clément-Fontaine (1999) portant sur la licence la plus populaire de l'open source : la GPL (General Public License).

référence aux éléments qui caractérisent à la fois : le processus de conception des logiciels libres : c'est-à-dire la description des différentes manières dont ces logiciels sont produits [Partie II et III] ; leur exploitation marchande : autrement dit la manière dont les firmes et les autres organisations génèrent des revenus grâce à ces logiciels [Partie IV] ; leurs propriétés notamment en matière d'utilisation et d'innovation [Partie V]. Mais au préalable, il convient de présenter comment des mutations industrielles ont conduit à la révision conceptuelle de certains objets en sciences de gestion [Partie I].

PARTIE I. L'OPEN SOURCE : DE LA MUTATION D'UNE INDUSTRIE A LA REVISION CONCEPTUELLE D'OBJETS EN SCIENCES DE GESTION.

Chapitre 1. LES REGIMES DE L'OPEN SOURCE : UNE SYNTHESE DES ENJEUX DE LA DIFFUSION DU LOGICIEL LIBRE.

1. Une synthèse des enjeux organisationnels, économiques et technologiques de la diffusion du logiciel libre : apports des régimes de l'open source.

Partie I : L'open source : de la mutation d'une industrie à la révision conceptuelle d'objets en sciences de gestion.

Le but de cette partie est d'exposer les enjeux et les problématiques liés au développement de l'open source qui sont au cœur de cette thèse.

Nous décrivons d'abord les mutations provoquées par l'open source ainsi que la méthodologie adoptée pour notre recherche [Chapitre 2]. Plus précisément l'open source a conduit à trois grandes révisions conceptuelles [1.]. Premièrement, l'apparition d'un nouveau type de solidarité compatible avec : une distribution géographique, une ouverture à d'autres que le groupe initial et une implication des entreprises [1.1.]. Deuxièmement, le rôle de la production non-marchande est renouvelé. En effet, dans l'industrie informatique le non-marchand a des liens très intenses et complexes avec le marché : d'un côté, certaines firmes utilisent le non-marchand comme un outil pour relancer la concurrence ; et d'un autre coté, certaines organisations à but non lucratif utilisent l'activité marchande pour remplir leurs objectifs [1.2]. Troisièmement, l'open source redéfinit la structure de l'industrie du logiciel en remettant en question le rôle des éditeurs, utilisateurs et intégrateurs ainsi que les rapports entre ces acteurs [1.3].

L'ampleur et l'intensité des changements provoqués par le logiciel libre fait émerger un ensemble d'interrogations synthétisées sous la forme de questions de recherche générales portant sur trois volets : l'organisation, la valorisation et l'innovation.

Volet 1 : l'organisation de l'open source.

- *QR 1 : D'où provient le modèle racine à l'origine du premier logiciel libre ?*
- *QR 2 : Quelles sont les expansions de ce modèle, sur quels critères distinguer les formes résultantes et comment suivre les transformations organisationnelles dans l'open source ?*

Volet 2 : la valorisation de l'open source.

- *QR 3 : Comment les logiciels libres sont-ils financés et valorisés ?*
- *QR 4 : Quel est l'impact de l'open source sur l'industrie du logiciel ?*

Volet 3 : l'innovation dans l'open source.

- *QR 5 : Quels types d'innovations les organisations de l'open source produisent-elles ?*
- *QR 6 : Quelle différence y a-t-il entre l'innovation dans les domaines du logiciel libre et du logiciel fermé ?*

Ensuite, nous décrivons les principales méthodes employées dans le cadre de notre recherche [2.]. En premier lieu, nous présentons notre approche historique pour caractériser des objets émergents mettant en défaut les connaissances existantes et pour analyser des cas sur une longue période [2.1.]. En second lieu, nous présentons les opportunités offertes par internet en termes de nouvelles techniques de recherche [2.2.]. D'une part, l'entretien par internet introduit une dynamique inédite des savoirs au cours des entretiens : l'apprentissage externe. Ce mode d'entretien permet de mobiliser des connaissances autres que celle du chercheur et du répondant pour générer des questions qu'il n'aurait pas été possible de poser sans internet. D'autre part, la création d'un collège d'experts en ligne basé sur la logique et l'esprit de la technique *« Delphi »* propose une méthode contingente et innovante pour étudier l'innovation dans l'industrie du logiciel. De manière beaucoup plus classique, nous avons réalisé des études de cas pouvant être classifiées en deux catégories : les cas majeurs et mineurs. D'un côté, les cas majeurs ont largement contribué à la construction des différentes parties de notre travail ; de l'autre, les cas mineurs y ont apporté une contribution beaucoup moins importante.

Dans le troisième chapitre, nous réalisons une revue de la littérature sur les organisations de l'open source [Chapitre 3]. Nous décrivons d'abord la littérature sur les communautés open source [1.]. Ce faisant nous centrons notre lecture autour de deux types de travaux : d'une part, ceux se focalisant sur l'aspect indépendant et distribué des acteurs [1.1.] ; d'autre part, ceux portant sur la gouvernance et la division du travail dans les communautés [1.2.].

Ensuite, nous étudions les travaux sur le rôle des firmes dans l'open source [2.]. Ces derniers sont regroupés en deux thèmes : d'un côté, l'étude des rapports entre firmes et communautés [2.1.] ; et de l'autre, la revue des quelques auteurs analysant les firmes comme créateurs de communautés [2.2.]. Globalement dans la littérature, l'aspect distribué des communautés est bien pris en compte. En revanche, l'état des connaissances sur les éléments de solidarité liant les participants des différentes organisations est particulièrement lacunaire. Peu de travaux existent également sur les organisations constituées majoritairement d'entreprises.

Partie II : Caractérisation et généalogie du modèle racine : la combinaison inédite entre un système de management de la solidarité et un système de production distribué.

L'objectif de cette partie est de caractériser le modèle racine à l'origine du premier logiciel libre. Pour cela, nous réalisons un détour historique.

Dans le premier chapitre, nous analysons les travaux sur la caractérisation des communautés afin de mettre en évidence leurs apports et limites ainsi que notre positionnement vis-à-vis de la littérature [Chapitre 1].

Les travaux sur la caractérisation des communautés se regroupent en trois cadres d'analyse : les communautés de pratiques et épistémiques, la *« collective invention »* et les réseaux coopératifs. Nous montrons d'abord que la communauté d'utilisateurs-développeurs n'est pas une communauté de pratique mais que celle-ci a toutefois une dimension épistémique [1.].

Puis, nous soulignons que la notion d'invention n'est pas adaptée pour nommer le phénomène décrit par Allen. En effet, il s'agissait d'un cas où des producteurs de fer au XIX^{ème} siècle partageaient les améliorations de performances de hauts-fourneaux utilisés comme outil de production. En outre, la littérature confirme que les communautés open source ont deux différences avec les cas historiques de *« collective invention »* : d'une part dans la *« collective invention »*, il y a une uniformisation des objectifs alors que dans les communautés open source les participants ont des objectifs divergents (Henkel 2006: 955) ;

d'autre part, Osterloh et Rota (2007) ont souligné que lorsqu'une technologie passait au stade de l'exploitation les groupes de *« collective invention »* disparaissaient alors que les communautés open source existent toujours en dépit du fait que le phénomène soit également passé au stade de l'exploitation [2.].

Enfin la littérature sur les réseaux de collaboration souligne que la production non-marchande a pris une ampleur particulièrement considérable et plus encore sur internet. Dans ce cadre, la production non-marchande en ligne a reçu plusieurs qualificatifs : elle est tantôt nommée *« peer-production »* (Benkler et Nissenbaum 2006), *« social production »* (Benkler 2006) ou *« réseau coopératif »* (Grassineau 2009). D'autre part, certains auteurs défendent l'idée que le modèle des communautés open source n'est pas un nouveau modèle de production du point de vue de historique (Dalle et al. 2005: 397) [3.]. Le cadre des réseaux coopératifs est le cadre avec lequel nous partageons le plus d'affinités.

Dans le deuxième chapitre, nous partons du constat qu'il n'existe pas de cadre analytique permettant de caractériser l'ensemble des systèmes de production marchands et non-marchands. En effet, il est nécessaire d'avoir un cadre dépassant la dichotomie hiérarchie/marché [Chapitre 2].

En premier lieu, nous faisons le lien entre deux corpus théoriques : d'un côté, la littérature en sciences sociales sur la solidarité et le don ; et d'un autre côté, la littérature en sciences économique et de gestion [1.]. Pour ce faire, nous introduisons en premier lieu la notion de solidarité pour laquelle nous présentons une généalogie des travaux de différents auteurs en sciences sociales ayant utilisé cette notion : More, Fourier et l'Ecole sociétaire, Tönnies, Durkheim, Bourgeois, Praca, Buchanan, et Mance [1.1.]. En nous inspirant des travaux sur le *« paradigme du don »*, nous suggérons un nouveau cadre capable d'intégrer l'ensemble des systèmes de production marchands et non-marchands : *le diptyque charité/marché* [1.2.].

Suite à cela, nous décrivons la variété des structures productives entre la charité et le marché [2.]. Il y a premièrement les structures se trouvant entre la charité et le donnant-donnant (la souscription) [2.2.] et celles représentant purement du donnant-donnant (la caisse d'épargne rotative) [2.3.]. Deuxièmement, il y a les formes historiques de la solidarité collective [3.] avec d'un côté les systèmes de solidarité à objet stable : cela signifie que l'on connaît à peu près ce qui est donné et ce qui est reçu en retour (l'Utopie, le Phalanstère et la Corporation) [3.1.] ; et d'un autre côté, il y a les systèmes de solidarité où l'objet est instable : c'est-à-dire que l'on sait ce que l'on doit donner mais on ignore ce que l'on va recevoir en retour (la Ghilde et la Tontine) [3.2.].

Ensuite, en nous basant sur ces cas historiques nous définissons la notion d'action collective solidaire [4.]. Puis nous appliquons cette notion pour décrire le modèle de la communauté d'utilisateurs-développeurs [5.]. Grâce à cette notion, nous détaillons les liens de solidarité liant les acteurs de cette structure : nous prouvons que la communauté d'utilisateurs-développeurs est un système de management de la solidarité particulier puisqu'il s'agit du seul système doté d'une solidarité ouverte sur l'extérieur du groupe [5.1]. Enfin, nous présentons nos apports vis-à-vis de la littérature [5.2.].

Dans le troisième chapitre, nous soulignons que les systèmes de production distribués ne sont pas nouveaux dans l'histoire des organisations et que les communautés open source n'ont pas l'exclusivité sur cela [Chapitre 3]. Nous nous intéressons d'abord à l'histoire des systèmes de production distribués [1.] : nous nous focalisons sur trois cas : la manufacture dispersée [1.1], l'industrie domestique [1.2] et la franchise [1.3].

Pour caractériser la communauté d'utilisateurs-développeurs [2.], nous identifions tout d'abord les éléments communs à l'ensemble des systèmes de production distribués [2.1.]. Plus précisément, ces systèmes ont deux caractéristiques communes : *la distribution des acteurs* et *l'agglutination de l'objet de production* [2.2.]. Ensuite, nous décrivons les éléments rendant possible la production distribuée de logiciels [2.3.] en décrivant l'influence de quatre facteurs : la création d'un nouvel objet virtuel aux propriétés particulières (le logiciel), le partage de langages de programmation, le développement de la micro-informatique et d'internet.

Grâce au travail réalisé dans cette partie, nous répondons à la première question de recherche : *D'où provient le modèle racine à l'origine du premier logiciel libre ?* (QR 1) par la première thèse de ce travail : *La communauté d'utilisateurs-développeurs s'appuie : d'une part sur un système de management de la solidarité ouvert où les individus sont unis par une solidarité de conception, d'apprentissage et d'usage ; d'autre part sur un système de production distribué nécessitant un faible investissement capitalistique et dont l'objet réalisé a des propriétés fortement agglutinantes* (TH 1).

Partie III : Du modèle racine à la variété des organisations : l'émergence des régimes de l'open source.

L'objectif de cette partie est de présenter les propriétés et expansions des régimes de l'open source sur la base d'un travail empirique réalisé sur une longue période.

Dans le premier chapitre, nous détaillons la construction des régimes de l'open source [Chapitre 1]. Tout d'abord, nous en présentons les fondements théoriques et pratiques. Pour ce faire, nous décrivons les cinq études de cas sélectionnées grâce à la méthode de l'échantillonnage théorique sur la base de quatre critères : le type d'acteurs, le mode de financement, le système de rémunération et la structure juridique [1.].

Ensuite, nous réalisons cinq monographies [2.] sur les cas suivants : Kexi, une communauté d'utilisateurs-développeurs assurant le développement d'un système de gestion de base de données avec une interface graphique [2.1.] ; Freeworks, une association de loi 1901 gérant différentes initiatives autour du concept de l'open source [2.2.] ; Mandriva, une société française éditant une distribution Linux et gérant une communauté d'utilisateurs-développeurs contribuant à ce logiciel [2.3.] ; OpenOffice.org, une communauté rassemblant principalement des firmes de l'industrie informatique développant une suite bureautique à environnements multiples [2.4.] ; OW2 Consortium, un consortium industriel rassemblant plus de 60 entreprises et instituts de recherche travaillant sur les problématiques du middleware [2.5.].

Dans le second chapitre, nous modélisons les régimes de l'open source dérivés des cas étudiés [Chapitre 2]. Dans un premier temps, nous définissons les paramètres organisationnels dominants des organisations étudiées [1.] grâce auxquels nous produisons le modèle du pentagone comprenant cinq formes organisationnelles [2.].

Puis dans un second temps, nous décrivons chaque idéal-type [3.] : *la firme de l'open source* [3.1.], *la communauté d'utilisateurs-développeurs* [3.2.], *l'association d'utilisateurs-développeurs* [3.3.], *la communauté inter-organisations* [3.4.] et *le consortium de l'open source* [3.5.].

Enfin dans un troisième temps, nous analysons les formes hybrides existantes de l'open source [4.]. Nous relisons les cas étudiés sous l'angle de l'hybridation [4.1.]. Puis, nous utilisons le modèle du pentagone pour détailler quelques formes hybrides [4.2.].

Suite aux réflexions de cette partie, nous apportons une réponse à la seconde question de recherche : *Quelles sont les expansions de ce modèle, sur quels critères distinguer les formes résultantes et comment suivre les transformations organisationnelles dans l'open source ?* (QR 2) par la seconde thèse de ce travail : *La communauté d'utilisateurs-développeurs est le*

modèle racine à l'origine de différents régimes de production celui-ci a connu différentes expansions (synthétisées dans le modèle du pentagone) dues à l'implication croissante des firmes dans le phénomène open source. (TH 1).

Partie IV : Les stratégies dans l'open source : des combinaisons nouvelles entre le marchand et le non-marchand.

Dans cette partie, nous étudions les stratégies des entreprises et des autres organisations impliquées dans l'open source.

Dans le premier chapitre, nous exposons les stratégies de valorisation économique des logiciels libres [Chapitre 1]. Tout d'abord, nous réalisons une revue de la littérature [1.] dans laquelle nous identifions deux types de stratégies : les stratégies ouvertes (basées sur de la prestation de services) [1.1.] et les stratégies hybrides (couplant services et licences) [1.2.]. Puis, nous synthétisons les modèles de valorisation des logiciels libres sur la base des différents travaux réalisés dans ce champ [1.3.]. Enfin, nous introduisons une troisième catégorie de modèles économiques reposant sur *des stratégies de verrouillage* [2.]. Ces stratégies insèrent un véritable paradoxe en faisant un bien privé à partir de ressources collectives, inversant en quelque sorte la logique du *« private-collective model »* (Stuermer, Sebastian, et von Krogh 2009; von Hippel et von Krogh 2003; von Krogh 2006). Plus précisément, trois stratégies sont identifiées : les stratégies d'intégration de logiciel libre dans du matériel [2.1.], les stratégies de *« forking*[26] *»* non libre et les stratégies basées sur une logique de services en ligne [2.2.].

Dans le second chapitre, nous nous intéressons aux stratégies industrielles et à la concurrence [Chapitre 2]. Premièrement, nous analysons la stratégie de Thales, un groupe industriel spécialisé en électronique et en systèmes d'information dans les domaines de la défense, l'aéronautique et la sécurité [1.]. Nous exposons comment la stratégie du groupe est passée d'une logique de développement fermée à une logique plus ouverte motivée par de réels enjeux technologiques et économiques [1.1.]. Ensuite, nous étudions l'implication de Thales dans l'open source [1.2.]. Nous scindons les activités du groupe en deux parties : les activités non stratégiques et stratégiques. Nous décrivons la manière dont l'industriel mutualise une partie de ses développements avec d'autres organisations dans divers

[26] Une stratégie de *« forking »* consiste à créer un logiciel libre sur la base d'un autre. Normalement, l'œuvre dérivée doit adopter la même licence que l'œuvre originale toutefois sur le terrain nous découvrons que certaines firmes développent différentes stratégies afin de *fermer* le logiciel ainsi créé.

consortiums et projets industriels pour des raisons d'indépendance technologique et économique. D'autre part, nous découvrons que Thales ne fait pas de valeur sur les logiciels libres en soi mais sur les parties *« hautes*[27] *»* que la firme ajoute aux logiciels libres qu'elle utilise comme composants.

Deuxièmement, nous nous penchons sur la dynamique coopérative et concurrentielle dans l'open source [2.]. En premier lieu, nous exposons les raisons ayant motivé nos investigations théoriques et empiriques sur la question de l'imbrication des connaissances [2.1.]. En second lieu, nous étudions deux cas (Thalix et Joram) afin d'offrir une vision pratique de ce phénomène [2.2.] En analysant ce matériau à travers le prisme de la théorie CK employée comme cadre analytique, nous modélisons le couplage entre le code source (SC) et les connaissances (K) issues de l'activité de développement. Grâce à cette modélisation, nous soutenons l'idée que des connaissances non génériques sont développées pendant le processus de conception permettant à des firmes de créer un modèle économique reposant sur une expertise. En somme, nous démontrons que la mise à disposition du code source d'un logiciel ne signifie pas forcément *libre concurrence* pour les services associés à cette technologie surtout lorsque le logiciel repose sur *un socle technologique*[28] complexe [2.3].

Dans le troisième chapitre, nous nous sommes intéressés à la dynamique des régimes de l'open source [Chapitre 3]. Dans un premier temps, nous explicitons ce concept [1.]. Puis dans un second temps, nous analysons la dynamique de deux cas d'un point de vue longitudinal. Premièrement, le cas OpenOffice.org illustre la manière dont le logiciel libre est employé par un ensemble d'entreprises afin de relancer certains marchés caractérisés par une domination monopolistique. [2.]. Deuxièmement, le cas Mozilla présente un cas singulier d'utilisation du marché pour financer des objectifs non-marchands. En outre, le cas Mozilla suggère également que la valorisation des logiciels orientés utilisateurs-finaux est difficile avec un modèle libre [3.].

Grâce au travail réalisé dans cette partie, nous suggérons des réponses aux deux questions de recherche portant sur la valorisation des logiciels libres : *Comment les logiciels libres sont-ils valorisés ?* (QR3) et *QR 4 : Quel est l'impact de l'open source sur l'industrie du logiciel ?*

[27] En informatique, la structure d'un système comprend trois parties : les couches basses (le matériel), les couches intermédiaires (le *« middleware »* ou intergiciel) et les couches hautes (les applications utilisateur-final ou métier).

[28] Un socle technologique est un ensemble de technologies pouvant être un standard industriel, une norme certifiée, un langage de programmation, etc. sur lequel un logiciel s'appuie dans une large mesure. Par conséquent, la maîtrise du logiciel nécessite également la connaissance de ce socle.

(QR4) par les deux thèses suivantes : *Les logiciels libres sont valorisés par des modèles : (1) ouverts, (2) hybrides et (3) fermés (basés sur des stratégies de verrouillage). Toutefois, les logiciels orientés utilisateur final sont difficilement valorisables avec un modèle ouvert. Seules trois possibilités sont empiriquement identifiées : le financement publicitaire, le modèle de la souscription ou la cession de licence (modèle hybride ou fermé)* (TH 3) et *Les logiciels libres sont utilisés comme des composants génériques sur la base desquels la valeur est créée dans l'industrie permettant l'indépendance technologique. Néanmoins leur pleine exploitation économique nécessite deux catégories de connaissances : l'une est générique (intégration), l'autre repose sur une expertise (édition)* (TH 4).

Partie V : L'innovation dans les régimes de l'open source : compenser les défaillances de l'offre marchande.

Dans cette partie nous étudions l'innovation produit et applicative dans les régimes libres et dans une moindre mesure dans les régimes fermés.

Dans le premier chapitre, nous suggérons un état de l'art sur la *« user innovation »* et sur l'évaluation de l'innovation dans l'open source [Chapitre 1]. Premièrement, nous retraçons la construction de la littérature sur les communautés open source comme un prolongement des travaux sur la *« user innovation »* [1.]. Tout d'abord, nous décrivons les travaux identifiant l'utilisateur comme une source d'innovations [1.1.]. Puis, nous exposons les résultats d'une récente recherche étudiant le rôle des utilisateurs dans la construction d'une offre radicalement innovante (Scheid 2007, 2009) [1.2.]. Et enfin, nous exposons les conditions de l'émergence d'une communauté d'utilisateurs indépendante [1.3.].

Deuxièmement, nous caractérisons les limites de la littérature existante sur l'évaluation des logiciels libres [2.]. Dans un premier temps, nous montrons qu'il existe une confusion entre processus innovant et innovation produit [2.1.]. Dans un second temps, nous analysons les recherches proposant une évaluation de l'innovation des logiciels libres [2.2.]. Dans un troisième temps, nous expliquons qu'il existe un manque de critères pertinents pour évaluer l'innovation dans l'industrie du logiciel [2.3.]. Pour finir, nous exposons la vision du logiciel libre de quelques praticiens [2.4.]

Dans le second chapitre, nous évaluons l'innovation produit des logiciels libres [Chapitre 2]. Primo, nous détaillons notre démarche de couplage de méthodologies [1.] dans laquelle nous exposons le problème de mesure de l'innovation rencontré [1.2.] et les différentes

techniques pour la mesurer [1.3.]. Secundo, nous présentons la logique du passage de la technique Delphi à la *« web-based Delphi »* [2.]. Tertio, nous détaillons cette approche [3.] : nous réalisons d'abord la définition qualitative de cinq types d'innovation [3.1], puis nous demandons à 125 experts géographiquement distribués d'évaluer 25 logiciels libres [3.2.], nous modélisons ensuite l'innovation logicielle en nous basant sur le jugement moyen d'experts [3.3.] et pour finir nous élargissons la procédure d'évaluation à 152 logiciels libres [3.4.]. Quarto, nous discutons des implications de notre recherche [4.] où nous soulignons qu'une grande partie des innovations produit conçues par des utilisateurs-développeurs sont orientées vers la résolution de problèmes.

Dans le troisième chapitre, nous étudions les modèles d'innovation libres et fermés [Chapitre 3]. Tout d'abord, nous décrivons ce que nous nommons l'innovation par les extrémités [1.]. Pour ce faire, nous proposons trois éléments : une présentation des bases théoriques et empiriques de ce modèle [1.1], une théorisation générique de l'histoire du développement des logiciels libres [1.2] et une révision du *« bazaar »* de Raymond [1.3]. Ensuite, nous étudions les enjeux de la conception de logiciels fermés [2.] comprenant d'une part, une confrontation de cas empiriques [2.1] et d'autre part, une théorie de la conception des logiciels [2.2]. Pour finir, nous revenons sur les enjeux de la conception de logiciels libres [3.] : d'un côté, créés par une communauté d'utilisateurs-développeurs [3.1] et de l'autre, initiés par une firme [3.2].

Suite aux différents travaux réalisés dans cette partie, nous sommes en mesure de répondre à la troisième catégorie de questions de recherche portant sur l'innovation : *Quels types d'innovations les organisations de l'open source produisent-elles ?* (QR 5) et *Quelle différence y a-t-il entre l'innovation dans les domaines du logiciel libre et du logiciel fermé ?* (QR 6) par les deux thèses suivantes : *La majorité des innovations conçues par des utilisateurs dans l'open source sont orientées vers la résolution de problèmes ; lorsqu'elles sont conceptuelles : elles sont concentrés dans trois domaines : la communication, le développement et l'administration de système* (TH 5) et *Les acteurs des modèles libre et fermé ont des visions de l'innovation différentes. D'un côté, les logiciels libres favorisent l'innovation par les extrémités tandis que les logiciels fermés privilégient l'innovation par les concepts* (TH 6).

Partie VI : Conclusion : apports, applications et implications des régimes de l'open source.

Pour conclure notre travail, nous résumons les apports de cette thèse sur les axes: organisationnel, économique et technologique [1.]. Concernant l'axe organisationnel, nous montrons que le modèle racine à l'origine du premier logiciel libre est issu de la convergence inattendue entre un système de solidarité et un système de production distribué. Ensuite, nous décrivons les nouvelles interactions entre l'activité marchande et non-marchande donnant naissance à des formes organisationnelles originales et inédites [1.1].

Du point de vue économique, nous identifions un nouveau type de modèle d'affaires consistant à utiliser divers mécanismes technologiques afin de contourner les règles des licences. Nous détaillons également la manière dont les logiciels libres sont utilisés comme composants génériques par des groupes industriels dans le but de générer de la valeur [1.2].

Enfin pour l'aspect technologique, nous suggérons que les innovations créées par les organisations de l'open source visent principalement à combler les défaillances du système marchand. Nous découvrons en outre que les acteurs du logiciel libre favorisent l'innovation par les extrémités (innovation applicative) tandis que les firmes du logiciel fermé privilégient l'innovation par les concepts.

Sur la base de l'ensemble de nos apports, nous détaillons un modèle d'aide à la décision à destination des gestionnaires de technologies logicielles [2.]. Ce modèle permet aux décideurs de rationnaliser le choix de l'ouverture ou de la fermeture d'une technologie sur la base des espaces de valeurs identifiés [2.1.]. Puis, nous suggérons des exemples d'application *ex-post* et *ex-ante* [2.2.].

Pour finir, nous nous penchons sur les implications managériales et théoriques de notre recherche notamment en discutant de l'actuel courant dominant de la littérature sur l'innovation : le paradigme de l'open innovation [3.]. Nous discutons deux idées : d'une part, de la complémentarité des logiques ouvertes et fermées [3.1.] ; d'autre part, du rôle central des activités de R&D dans la construction de l'avantage concurrentiel des firmes dans l'open source [3.2].

2. Les principaux résultats et méthodologies.

Le tableau ci-dessous résume les apports de notre recherche.

Tableau 2 : Résumé des questions de recherche et des thèses.

Volet 1 : l'organisation.	QR 1 : D'où provient le modèle racine à l'origine du premier logiciel libre ?	Thèse 1 : La communauté d'utilisateurs-développeurs s'appuie : d'une part sur un système de management de la solidarité ouvert où les individus sont unis par une solidarité de conception, d'apprentissage et d'usage ; d'autre part sur un système de production distribué qui nécessite un faible investissement capitalistique et dont l'objet réalisé à des propriétés fortement agglutinantes.
	QR 2 : Quelles sont les expansions de ce modèle, sur quels critères distinguer les formes résultantes et comment suivre les transformations organisationnelles dans l'open source ?	Thèse 2 : La communauté d'utilisateurs-développeurs est le modèle racine à l'origine de différents régimes de production celui-ci a connu différentes expansions (synthétisées dans le modèle du pentagone) dues à l'implication croissante des firmes dans le phénomène open source.
Volet 2 : la valorisation.	QR3 : Comment les logiciels libres sont-ils valorisés ?	Thèse 3 : Les logiciels libres sont valorisés par des modèles : (1) ouverts, (2) hybrides et (3) fermés (basés sur des stratégies de verrouillage). Toutefois, les logiciels orientés utilisateur final sont difficilement valorisables avec un modèle ouvert. Seules trois possibilités sont empiriquement identifiées : le financement publicitaire, le modèle de la souscription ou la cession de licence (modèle hybride ou fermé).
	QR 4 : Quel est l'impact de l'open source sur l'industrie du logiciel ?	Thèse 4 : Les logiciels libres sont utilisés comme des composants génériques sur la base desquels la valeur est créée dans l'industrie permettant l'indépendance technologique. Néanmoins leur pleine exploitation économique nécessite deux catégories de connaissances : l'une est générique (intégration), l'autre repose sur une expertise (édition).
Volet 3 : l'innovation.	QR 5 : Quels types d'innovations les organisations de l'open source produisent-elles ?	Thèse 5 : La majorité des innovations conçues par des utilisateurs dans l'open source sont orientées vers la résolution de problèmes ; lorsqu'elles sont conceptuelles : elles sont concentrés dans trois domaines : la communication, le développement et l'administration de système.
	QR 6 : Quelle différence y a-t-il entre l'innovation dans les domaines du logiciel libre et du logiciel fermé ?	Thèse 6 : Les acteurs des modèles libre et fermé ont des visions de l'innovation différentes. D'un côté, les logiciels libres favorisent l'innovation par les extrémités tandis que les logiciels fermés privilégient l'innovation par les concepts.

Tableau 3 : Synthèse des méthodologies.

Parties	Chapitres	Méthodologies
PARTIE II. CARACTERISATION ET GENEALOGIE DU MODELE RACINE : LA COMBINAISON INEDITE ENTRE UN SYSTEME DE MANAGEMENT DE LA SOLIDARITE ET UN SYSTEME DE PRODUCTION DISTRIBUE.	CHAPITRE 2. A LA RECHERCHE DES RACINES DE L'OPEN SOURCE (1) : RETOUR SUR L'HISTOIRE DE L'ACTION COLLECTIVE SOLIDAIRE.	- Approche historique. - Etude de cas classique. - Echantillonnage théorique.
	CHAPITRE 3. A LA RECHERCHE DES RACINES DE L'OPEN SOURCE (2) : RETOUR SUR LES FORMES DE PRODUCTION DISTRIBUEES.	- Approche historique. - Etude de cas classique. - Echantillonnage théorique.
PARTIE III. DU MODELE RACINE A LA VARIETE DES ORGANISATIONS : L'EMERGENCE DES REGIMES DE L'OPEN SOURCE.	CHAPITRE 1. DE LA VARIETE DES ORGANISATIONS OU L'EMERGENCE DES REGIMES DE L'OPEN SOURCE.	- Etude de cas classique. - Etude de cas longitudinale. - Entretien en ligne. - Entretien par téléphone. - Entretien en face à face. - Echantillonnage théorique.
	CHAPITRE 2. LA MODELISATION DES REGIMES DE L'OPEN SOURCE.	- Etude de cas classique. - Etude de cas longitudinale. - Entretien en ligne. - Entretien par téléphone. - Entretien en face à face. - Echantillonnage théorique.
PARTIE IV. LES STRATEGIES DANS L'OPEN SOURCE : DES COMBINAISONS NOUVELLES ENTRE LE MARCHAND ET LE NON-MARCHAND.	CHAPITRE 1. VALORISATION ECONOMIQUE DE L'OPEN SOURCE ET STRATEGIES.	- Etude de cas classique. - Entretien en ligne. - Entretien par téléphone. - Entretien en face à face. - Echantillonnage théorique.
	CHAPITRE 2. STRATEGIE INDUSTRIELLES ET CONCURRENCE.	- Etude de cas classique. - Entretien en ligne. - Entretien par téléphone. - Entretien en face à face.
	CHAPITRE 3. ANALYSE DYNAMIQUE DES REGIMES DE L'OPEN SOURCE.	- Approche historique. - Etude de cas longitudinale. - Entretien en ligne.
PARTIE V. L'INNOVATION DANS LES REGIMES DE L'OPEN SOURCE : COMPENSER LES DEFAILLANCES DE L'OFFRE MARCHANDE.	CHAPITRE 2. DE LA METHODE DELPHI A LA CONSTRUCTION DE LA WEB-BASED DELPHI : ETUDE EMPIRIQUE DE L'INNOVATION DANS LES COMMUNAUTES D'UTILISATEURS-DEVELOPPEURS.	- Etude de cas classique. - *« Web-based Delphi »* - Enquête par questionnaire. - Entretien en ligne. - Entretien par téléphone. - Entretien en face à face. - Analyses quantitatives (analyse en composantes principales, catégorisation statistique).
	CHAPITRE 3. MODELES D'INNOVATION LIBRES ET FERMES.	- Approche historique. - Etude de cas longitudinale. - Echantillonnage théorique. - Entretien en ligne. - Entretien par téléphone - Entretien en face à face.

Chapitre 2. PROBLEMATIQUE ET METHODOLOGIE.

1. La diffusion du logiciel libre : révisions conceptuelles et questions de recherche.

Le logiciel libre a bouleversé bon nombre de notions que l'on imaginait jusqu'ici stabilisées. Il a remis en question : la représentation de la solidarité **[1.1.]**, le rôle de la production non marchande **[1.2.]** et la structure de l'industrie du logiciel **[1.3.]**. Ces redéfinitions conduisent à la définition d'un ensemble de questions de recherche au centre de cette thèse.

1.1. Une nouvelle forme de solidarité.

La notion de solidarité en sciences sociales désigne un élément liant des individus d'un collectif. Historiquement, la notion de solidarité s'est développée parmi des individus ayant une proximité géographique que se soit dans la littérature sur les *communautés utopiques* (Fourier 1845a, 1845b; More 1842, 1987; Tönnies 1922), les formes historiques de la solidarité (Ghilde, Corporation, Tontine) ou plus récemment en économie sociale et solidaire (Ardener 1964; Gasse-Hellio 2002; Mance 2003; Vanneuville-Zerouki 1999). Or, l'open source a donné naissance à des collectifs d'acteurs, de plus ou moins grande taille, géographiquement distribués liés par des liens de solidarité particuliers[29].

Ensuite, la solidarité est souvent synonyme d'économie autarcique avec peu d'interaction avec l'extérieur du groupe solidaire et encore moins d'impact marchand. En revanche, dans le cadre de l'open source, les communautés produisent des logiciels qui d'une part peuvent être utilisés par tous ceux qui le souhaitent (en respectant la licence d'utilisation) et d'autre part peuvent servir pour une valorisation plus ou moins directe sur le marché.

Enfin, la notion de solidarité est normalement employée pour décrire les liens unissant des individus. Pourtant, dans l'open source il existe des collectifs d'entreprises et plus

[29] Nous présenterons ces trois dimensions de la solidarité dans les communautés d'utilisateurs-développeurs de manière beaucoup plus approfondie dans le chapitre dédié à l'étude de l'action collective solidaire.

généralement d'organisations liées par des liens de solidarité comme en témoigne le cas OW2.

Encadré 1 : le cas OW2.

OW2 est une association de loi 1901 régie par ses propres organes de direction. Le consortium rassemble plus de 60 organisations de l'industrie informatique (Thales, France Telecom, Bull, etc.) et de la recherche (Focus, INRIA, CNRS, etc.) visant à développer une base technologique middleware en open source. OW2 est aussi un lieu où les collaborations industrielles sont abondantes.

OW2 n'est pas soumise aux lois anti-concentration car la production du consortium est publique : n'importe qui peut télécharger du code produit par OW2 sans aucune discrimination.

OW2 est issue de la fusion entre d'une part le consortium chinois OrientWare et d'autre part le consortium européen ObjectWeb. L'Europe et la Chine partagent toutes deux un grand intérêt pour le logiciel libre principalement du fait de la nature du tissu industriel de ces pays plutôt caractérisé par le métier d'intégration.

1.2. Le nouveau rôle de la production non marchande.

En sciences économiques, l'activité marchande est considérée comme la normalité. Cependant, le logiciel libre remet en question la vision *non économique* de la production dite *« non-marchande »*. Les logiciels libres sont produits par des organisations mélangeant bénévolat, travail salarié et travail financé. Le marchand est tantôt le but de l'organisation (firmes) tantôt il ne constitue qu'un moyen de financement de la production. Le cas Mozilla est particulièrement caractéristique de cette seconde vision de l'activité marchande.

Encadré 2 : Le cas Mozilla.

Mozilla est un ensemble de technologies internet dont les premiers développements sont issus de la recherche publique américaine. La firme Netscape fût créée en 1994 par des anciens de la NCSA (National Center for Supercomputing Applications).

En 1998, face à des difficultés économiques, Netscape décide de libérer le code source des technologies Mozilla et de créer une communauté composée principalement d'employés (la Mozilla Organization).

En 2003, un changement de stratégie est opéré au sein de Netscape qui provoque l'abandon des technologies Mozilla. C'est à partir de ce moment, qu'un nouveau type d'acteur entre en jeu, il s'agit de la Mozilla Fondation constituée par des anciens employés de Netscape.

La Mozilla Fondation n'a pas de but lucratif et utilise la sphère marchande pour s'autofinancer. En 2005, la Mozilla Fondation créa la Mozilla Corporation, une entité à but lucratif au service du non-marchand. En 2008, la Mozilla Fondation créé une nouvelle filiale du même type cette fois pour financer les technologies Thunderbird.

1.3. La mutation de l'industrie du logiciel.

Le logiciel libre a provoqué également une redéfinition de la figure même du logiciel. Un logiciel libre peut être exécuté, étudié, modifié et redistribué en respectant la licence régissant les règles de sa diffusion. Historiquement, l'industrie du logiciel en tant qu'entité autonome s'est fondée sur une logique de création de valeur reposant essentiellement sur l'usage du logiciel.

Le logiciel libre a quant à lui ouvert d'autres espaces de valeurs qui jusqu'ici étaient soit réservés aux éditeurs seuls soit n'existaient pas. Le logiciel libre a en quelque sorte restauré la logique première de l'industrie informatique où les logiciels étaient accompagnés de leur code source. Toutefois, une différence fondamentale existe entre la situation technologique de l'industrie informatique des années 70 et celle d'aujourd'hui. D'un côté dans les années 70, il y avait une hétérogénéité des systèmes d'exploitation ou plateformes d'accueil du logiciel rendant difficile le portage direct d'un système à l'autre et de ce fait le métier d'éditeur de logiciel n'avait pas encore de sens. D'un autre côté avec l'apparition de standards de fait, il y a une normalisation des infrastructures (PC) et des systèmes (Windows) permettant la standardisation des usages et par conséquent une économie industrielle du logiciel où conception et utilisation font l'objet d'activités distinctes. En d'autres termes, il s'agit de la relation entre l'éditeur concevant un produit et les clients se contentant de l'utiliser.

En outre, la nature du code informatique, la diffusion de langages de programmation, la démocratisation de la micro-informatique et d'internet, ont rendu possible la conception distribuée de logiciels par des groupes plus ou moins importants d'*utilisateurs experts*[30]. C'est l'image même de l'éditeur qui s'est trouvée bouleversée : du fait de la diffusion du code source des logiciels, l'activité d'édition n'était plus limitée au concepteur initial du logiciel. C'est également l'image cristallisée de l'utilisateur qui est défiée : l'utilisateur devient acteur de l'activité de conception[31]. Cette révision provoque une crise de légitimité de l'activité d'édition qui ne peut plus à elle seule valoriser certains types de logiciels. D'autre part, le métier de l'intégrateur est aussi profondément modifié : il devient responsable de briques technologiques et semble moins dépendants des éditeurs.

[30] Nous employons l'expression utilisateurs experts car originellement, le logiciel libre a émergé dans la sphère académique où les utilisateurs étaient aussi développeurs. Et comme le souligne un de nos interlocuteurs : *« l'open source c'est entre développeurs. L'utilisateur, lui il s'en [moque] du source… »*, Développeur-Musicien, Freeworks, entretien en face à face avec l'auteur, 5 juin 2006.

[31] Cet élément explique en partie le regain d'intérêt pour les travaux de von Hippel sur le rôle des utilisateurs à fin des années 80 (von Hippel 1988).

Le logiciel libre a aussi un impact *géotechnologique* et implique la souveraineté technologique des Etats et des entreprises. Il convient de signaler que l'activité d'édition est largement dominée par des groupes industriels américains. Le logiciel libre permet ainsi une remise en question de cette domination. L'Europe dispose de très peu d'éditeurs ayant une envergure internationale. La zone Europe est plutôt caractérisée par le métier d'intégrateur ceci explique l'intérêt européen pour les technologies open source.

Pour ce qui est de l'indépendance technologique à proprement parler, le Ministère de la Défense français s'est fortement intéressé aux logiciels libres car ils permettent de reprendre le contrôle de certaines parties du système d'information jugées critique ne pouvant pas rester raisonnablement sous une domination étrangère. D'autres Etats s'intéressent également au logiciel libre : la Chine développe des compétences importantes en matière de logiciels libres en participant à différentes initiatives open source (OW2 Consortium, QualiPSo). Du point de vue de l'entreprise, certaines firmes intègrent systématiquement des logiciels libres comme en témoigne l'encadré ci-dessous sur la stratégie de Thales.

Encadré 3 : le cas Thales.

Thales est un groupe industriel spécialisé dans l'électronique et les systèmes dans les domaines de la défense, la sécurité et l'aéronautique. Thales opère dans des secteurs où la criticité du système d'information est un élément clé.

Dans un premier temps, la firme a été confrontée à des problèmes de représentation de la sécurité comme une *boîte noire* où le logiciel libre entrait en contradiction du fait de son ouverture.

Toutefois, Thales a adopté des composants libres dans de nombreux projets industriels et a montré que la sécurité et l'open source n'étaient pas antinomiques. Aujourd'hui, Thales utilise massivement du logiciel libre pour créer des systèmes d'informations et la stratégie du groupe ne fait qu'aller dans le sens de l'utilisation de composants libres.

Face à l'ampleur des révisions conceptuelles mentionnées précédemment, un grand nombre de questions peuvent être dégagées. Toutefois ces questions peuvent être synthétisées sous la forme de questions de recherche (QR) générales regroupées en trois volets : un volet organisation, un volet valorisation et un volet innovation.

Volet 1 : l'organisation.

- *QR 1 : D'où provient le modèle racine à l'origine du premier logiciel libre ?*
- *QR 2 : Quelles sont les expansions de ce modèle, sur quels critères distinguer les formes résultantes et comment suivre les transformations organisationnelles dans l'open source ?*

Volet 2 : la valorisation.

- *QR 5 : Comment les logiciels libres sont-ils valorisés ?*
- *QR 6 : Quel est l'impact de l'open source sur l'industrie du logiciel ?*

Volet 3 : l'innovation.

- *QR 3 : Quels types d'innovations les organisations de l'open source produisent-elles ?*
- *QR 4 : Quelle différence y a-t-il entre l'innovation dans les domaines du logiciel libre et du logiciel fermé ?*

2. Méthodologie : le web comme instrument de recherche.

Du fait de la complexité de l'open source et de par sa nature pluridisciplinaire, nous avons choisi de ne pas nous cantonner à une seule méthodologie comme la plupart des travaux sur le sujet. Par exemple, O'Mahony analyse l'open source en tant que mouvement social en se basant sur une approche qualitative utilisant des méthodes ethnographiques (O'Mahony 2002: 54). Nous avons au contraire adopté *« une démarche d'exploration »* (Segrestin 2006: 16) couplant méthodologies qualitatives et quantitatives. Bien que ces méthodes soient souvent considérées comme opposées, nous avons tenté de dépasser le débat de méthode. Le lecteur doit garder à l'esprit que cette recherche fut menée principalement sur des organisations géographiquement distribuées et qu'il était nécessaire d'utiliser les nouvelles possibilités d'investigation proposées par internet pour étudier ce terrain. C'est la raison pour laquelle nous avons adapté différentes méthodes existantes à l'étude des communautés sur internet que nous présenterons dans les prochaines sous-parties.

2.1. L'histoire au service de la recherche en gestion.

Plusieurs chapitres de cette thèse sont basés sur l'approche historienne largement établie en sciences de gestion (de Vaujany 2007: 2). Nous avons travaillé sur la généalogie des formes de l'action collective solidaire et distribuée. Nous avons également employé cette méthode pour analyser des études de cas de manière dynamique. Ainsi toutes nos réflexions portant sur la dynamique des régimes de l'open source sont basée sur cette logique. Notre travail s'est largement appuyé sur les enseignements en épistémologie et en histoire des

systèmes industriels d'Armand Hatchuel (2005, 2007). Dans les deux sous-parties suivantes nous détaillerons notre démarche méthodologique utilisant l'approche historienne.

2.1.1. La méthode historienne : un outil adapté pour analyser l'émergence de nouveaux objets de gestion.

L'histoire peut être utilisée afin de décrire des formes nouvelles d'objets de recherche en se basant sur la généalogie des objets auxquels ils semblent se rattacher.

Les communautés d'utilisateurs-développeurs ne sont pas vraiment rattachables à une forme d'organisations déjà théorisée. C'est pourquoi nous avons choisi de partir à la recherche des racines historiques des communautés d'utilisateurs-développeurs pour identifier ce qu'elles ont de singulier et nouveau[32]. Dans ce cas, l'approche historienne nous est apparue adaptée. Nous avons essayé de voir quels éléments étaient caractéristiques des communautés d'utilisateurs-développeurs : nous nous sommes rapidement aperçus que deux notions étaient particulièrement adaptées pour décrire ces organisations : *la solidarité* et *la distribution*.

Les organisations basées sur la notion de solidarité telle que nous la définissons n'ont pas vraiment, du moins à notre connaissance, fait l'objet d'une étude approfondie donnant lieu à un modèle de l'organisation solidaire. Les organisations distribuées ont, quant à elles, fait l'objet de récentes études suite à l'émergence de communautés développant des logiciels libres. Toutefois, les notions de conception et de production distribuées ne sont pas nouvelles dans l'histoire des organisations.

La démarche méthodologique la plus adaptée pour retrouver les *racines perdues* d'une forme organisationnelle reste semble-t-il la méthode dite *historienne*. On pourrait se contenter de décrire l'objet de recherche à partir de la grammaire des formes organisationnelles existantes dans les manuels de théorie de organisations mais cela ne nous semblait pas du tout pertinent car les termes, les modèles, etc. de la littérature en théorie des organisations ne sont pas vraiment adaptés pour décrire ces objets émergents. Par exemple, il existe une hiérarchie dans bon nombre de communautés de l'open source mais cette hiérarchie n'est pas *statutaire* ou basée sur un lien de subordination, un contrat ou un autre moyen classique d'organiser le travail (Lefebvre 2003) : il s'agit plutôt d'un type de hiérarchie basé sur les compétences, une forme hybride entre le leadership et la hiérarchie telle que nous la connaissons.

[32] Notre démarche méthodologique est basé sur la logique proposée par David et Hatchuel (2007) où les auteurs suggèrent que l'essence de la recherche en management est de produire de nouveaux modèles d'action collective.

La communauté d'utilisateurs-développeurs est une forme qui a récemment émergé grâce à la convergence de quatre éléments[33] : les propriétés du code informatique, l'uniformisation des langages de programmation, le développement de l'ordinateur personnel et la création du réseau des réseaux. Toutefois le modèle de la communauté d'utilisateurs-développeurs a des liens historiques avec des formes beaucoup plus anciennes que sont les systèmes de management de la solidarité et les systèmes de production distribués.

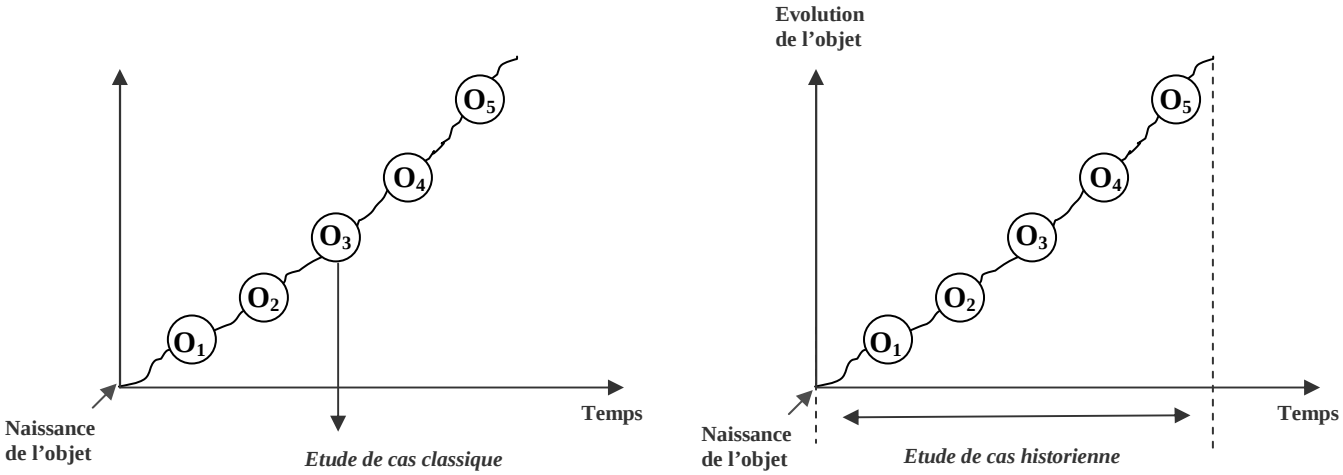
Dès lors, nous avons donc étudié les systèmes de production humains où la notion de solidarité était présente. Petit à petit, nous avons identifié un nombre substantiel d'organisations où la notion de solidarité était non seulement présente mais c'était véritablement la *raison d'être* de ces organisations. Nous nous sommes également intéressés aux systèmes de production distribués, nous avons vu que ces derniers avaient une histoire dissociable d'internet. Bien évidemment notre travail n'est pas une thèse en histoire mais en gestion empruntant des méthodes de l'historien. Notre démarche est donc largement basée sur des ouvrages, des recueils historiques plus que des archives de première main.

2.1.2. De l'étude de cas classique à l'étude de cas historienne.

Contrairement à la technique de l'étude de cas classique consistant à étudier une organisation à un instant *t* ou sur une très courte période, l'étude de cas historienne consiste notamment à suivre l'évolution d'organisations sur une longue période en partant le plus souvent de la naissance de l'objet. Cette méthode a largement été utilisée par Chandler dans ses travaux sur l'émergence de la grande entreprise (Chandler 1989), par de Vaujany sur le rôle de l'Eglise et de ses enclaves dans la formation et la diffusion de la gestion (de Vaujany 2007: 10) ou encore par Acquier dans son travail sur l'émergence de la Global Reporting Initiative ou GRI (Acquier 2007). Le schéma ci-dessous propose une représentation graphique matérialisant les différences entre l'étude de cas classique et historienne.

[33] Nous reviendrons plus longuement sur ces quatre éléments un peu plus loin dans notre travail.

Figure 1 : Comparatif entre l'étude de cas classique et historienne.



Le raisonnement historique fut employé dans différents chapitres et plus précisément pour étudier l'évolution des organisations de l'open source. Par exemple, bien que beaucoup de logiciels libres aient été initiés à l'origine par un développeur seul, les logiciels libres ont une origine hétérogène. Bon nombre de logiciels libres ont été initiés par des centres de recherche (exemple : le serveur Apache ou Mosaic) ou encore par des entreprises (exemple : MySQL, OpenOffice.org). La méthode historique fut donc utilisée pour suivre l'évolution des communautés de l'open source. Ce raisonnement nous a permis de retracer l'histoire de communautés d'un point de vue original et inédit dans la littérature.

En effet, l'histoire permet de voir les objets de manière dynamique au lieu de décrire un objet à un instant *t*. Etudier de manière statique des objets a certains avantages comme la profondeur de l'analyse mais cette méthode a aussi ses limites.

De nombreuses études ont été réalisées sur les communautés de l'open source néanmoins la littérature s'est focalisée sur l'étude statique de ces dernières ratant ainsi une large partie du phénomène. La littérature (West et O'Mahony 2005) a mis en évidence le fait qu'il était nécessaire d'avoir un modèle capable de rendre compte de la dynamique des organisations de l'open source, c'est ce que la méthode historienne apporte.

2.2. Les méthodes d'investigation par internet : une nouvelle forme d'immersion.

Les communautés de l'open source sont des organisations distribuées grâce aux technologies internet. De ce fait, l'étude de ces organisations nécessite une adaptation des

techniques d'investigation et même de *l'innovation méthodologique* afin d'être étudiés de manière satisfaisante.

Ce que nous essayerons de montrer dans cette section, c'est qu'internet ouvre la possibilité de créer de nouvelles méthodes de recherche. Dans la littérature sur l'open source, internet est souvent utilisé pour collecter des informations. Ainsi, O'Mahony a collecté notamment des données provenant d'archives de listes de diffusion (O'Mahony 2002: 59). Lee et Cole ont utilisé les archives des courriels du noyau Linux comme *« une source clé de données »* (Lee et Cole 2003: 637). von Krogh et al. ont aussi employés les archives publiques des courriers électroniques échangés par les membres du projet Freenet (von Krogh, Haefliger, et Spaeth 2003b: 1219). Pour n'en citer que quelques exemples.

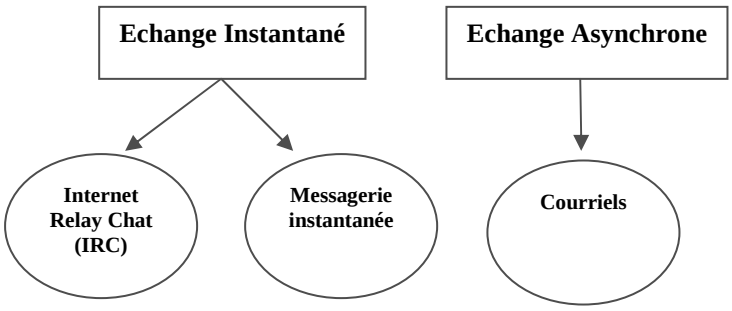
En revanche, internet est rarement exploité comme le support de nouvelles méthodes de recherche comme le propose les travaux de Kozinets à travers la notion de *« netnography »* (Kozinets 1997, 1998, 2002, 2006; Kozinets, Hemetsberger, et Schau 2008). La netnography est basée sur les méthodes de l'anthropologie culturelle (Kozinets 1997: 470). Kozinets définit la netnography comme *« une méthode interprétative créée spécifiquement pour étudier le comportement consommateur des cultures et des communautés présentes sur internet. »* (Kozinets 1998: 366). Plus simplement, la netnography, c'est l'utilisation de l'anthropologie en ligne (Kozinets 2006: 279). D'après Kozinets, cette méthode peut être employée de trois différentes manières : comme méthodologie, outil méthodologique ou outil exploratoire (Kozinets 1998: 367). Elle dépasse le simple téléchargement de *« posts*[34] *»* sur des forums (Kozinets 2006). La netnography nécessite une pleine participation du chercheur dans la culture étudiée (Kozinets 1998: 366; 2006: 286). Dans cette partie, nous ferons comme Kozinets, nous tenterons de mettre en exergue les potentialités d'internet en matière de nouvelles méthodes de recherche.

2.2.1. L'entretien via internet.

Pour résumer, nous avons utilisé trois moyens pour réaliser nos entretiens via internet. Ces différents moyens sont catégorisables en deux : ceux en échange instantané (IRC et messagerie instantanée) et ceux en échange asynchrone (courriels). Ci-dessous le schéma résume ces différents moyens.

[34] Un *« post »* est un message publié sur un forum.

Figure 2 : Les moyens permettant de réaliser des entretiens en ligne.



A. Les entretiens par échanges instantané.

→ L'entretien par IRC[35].

Les développeurs de l'open source utilisent différents moyens de communication pour coordonner leur travail mais aussi pour discuter des évolutions. L'IRC, est un outil utilisé par la quasi-totalité des organisations étudiées (OpenOffice.org, Kexi, Mandriva, etc.). Nous nous sommes connecté sur les canaux IRC de différentes organisations, nous y avons observé la manière dont les participants s'exprimaient et nous avons eu l'idée de réaliser quelques entretiens de groupe par ce biais.

L'entretien de groupe par IRC est un type d'entretien intéressant pour plusieurs raisons. Tout d'abord, l'IRC est un endroit particulièrement intéressant pour observer *in vivo* le travail des développeurs. Ce type d'entretien est particulièrement adapté lorsqu'il s'agit d'acteurs géographiquement distribués. De plus, l'IRC permet de réaliser des entretiens très longs alors que la norme des entretiens est beaucoup plus réduite. Certains de nos entretiens par IRC ont duré plus de 5 heures (soit 300 minutes). Dans le tableau ci-dessous nous présentons la moyenne des durées d'interviews réalisées dans quelques études dans le domaine de l'open source.

Tableau 4 : Durée moyenne des interviews dans quelques études sur l'open source.

Nombre d'interviews	Durée moyenne	Source
23	79 minutes	(Stuermer et al. 2009: 174)
25	45 minutes	(Alexy et Henkel 2007: 13)
30	53 minutes	(Gruber et Henkel 2006: 361)
45	90 minutes	(Shah 2006: 1003-04)
6	75 minutes	(West et O'Mahony 2007: 10)

[35] L'Internet Relay Chat ou IRC est un espace de discussion instantané utilisant des canaux de discussion.

→ **L'entretien par messagerie instantanée.**

L'entretien par messagerie instantanée consiste à utiliser un système de messagerie (Google Talk, Jabber, ICQ, etc.) comme support pour réaliser un entretien individuel. Certains de nos interlocuteurs ont préféré l'entretien en ligne à l'entretien par téléphone. Un ancien dirigeant décrivait de la manière suivante les membres de son entreprise : *« […] ils n'aiment pas trop se téléphoner, car ils n'aiment pas trop se parler*[36]*… »*.

B. Les entretiens par échanges asynchrone.

Le courrier électronique a largement été utilisé pendant dans notre recherche. Nous avons ainsi pris contact avec des personnes impliquées dans l'open source. Le courriel devient rapidement difficile à lire lorsqu'il y a trop de d'échanges. Nous avons donc privilégié le courriel lorsque nous avions des questions précises à poser. Au fur et à mesure, nous avons réussi à bâtir un véritable *réseau d'experts* que nous avons ensuite mobilisé lorsque nous avons jugé opportun de réaliser une étude sur l'évaluation de l'innovation[37].

C. L'entretien en ligne et l'apprentissage externe : une dynamique inédite des savoirs.

L'entretien en ligne a un avantage par rapport à l'entretien classique en face à face : il s'agit de la possibilité de rechercher des informations au cours de l'entretien. Traditionnellement, lorsque l'on ne connait pas un domaine ou que l'on ne comprend pas une notion dans un entretien, nous avons uniquement la possibilité de demander à notre interlocuteur de nous l'expliquer et par conséquent lui couper la parole. Sinon, il est difficile de continuer l'entretien. Lorsque l'entretien est réalisé en ligne, c'est différent : nous avons la possibilité de consulter une source externe sans couper, ni même que notre interlocuteur sache que nous sommes en train de rechercher des informations sur ce qu'il vient de dire. Cet apprentissage au cours de l'entretien permet donc de poser des questions que l'on n'aurait pas posées si l'entretien avait été réalisé par le biais d'un autre moyen de communication. Classiquement, un entretien est caractérisé par une double dynamique : celle des savoirs et celle des relations[38] (Hatchuel 2005). De manière simplifiée, il y a d'un côté les questions posées et de l'autre les réponses données. Lors d'un entretien le savoir acquis au cours de l'interview est toujours une fonction du savoir de l'interlocuteur et des connaissances actuelles du chercheur.

[36] François Bancilhon, ancien Directeur général, Mandriva, entretien en face à face avec l'auteur, mercredi 22 mai 2006.
[37] A ce sujet voir notre développement sur la technique du *« web-based Delphi »*.
[38] La distinction entre savoirs et relations est empruntée à Armand Hatchuel.

Soit : S_C = Savoir du chercheur ; S_I = Savoir de l'interlocuteur ; S_A = Savoir acquis. Par conséquent le savoir acquis peut être exprimé de la manière suivante :

$$S_A = f(S_C , S_I).$$

Le type de questions posées découlera logiquement des connaissances dont le chercheur dispose au moment où il pose sa question.

Avec l'*e-entretien*[39], c'est différent puisque le chercheur a la possibilité de mobiliser des savoirs autres que ceux dont il dispose (S_C) ou que son interlocuteur détient (S_I). Soit S_X : les connaissances externes. Par conséquent, le savoir acquis sera exprimé de la manière suivante :

$$S_A = f(S_C , S_I , S_X).$$

Grâce à cette nouvelle source de connaissances externes, le chercheur sera en mesure de poser des questions qu'il aurait été incapable de poser autrement. Nous illustrerons le phénomène de *l'apprentissage externe* par le biais de deux exemples au cours d'entretiens que nous avons mené en ligne (cf. encadré).

Encadré 4 : Exemples d'apprentissage externe.

Exemple 1 : Lors d'un de nos entretiens par le biais de l'IRC, nous ne savions pas ce que signifiait *Database FrontEnd* en informatique, un des membres nous a expliqué ce que cela signifiait et nous a renvoyé vers un lien.

<Fondateur 2> Nordine: *"I remember no details but originally it was planned as a database frontend"*
<Nordine> Fondateur 2 : *"And what is a database frontend?"*
<Fondateur 2> Nordine: *"a GUI app that helps you administer and design databases, something closely tied to the server databases. Example:"*
<Fondateur 2> : "http://en.wikipedia.org/wiki/Frontend"

Exemple 2 : Dans un autre entretien par courriel cette fois, nous avions demandé ce qu'était l'*ODF Alliance*.

Sophie G. : *« Il y a des projets comme l'ODF Alliance dans laquelle nous sommes engagés ? »*
[...]
Nordine : *« Que signifie l'ODF Alliance ? »* [...]
Sophie G. : *« ODF Alliance : http://formats-ouverts.org/blog/2006/03/03/727-voici-l-odf-alliance OpenOffice.org ayant servi dans ses premières versions au développement du format ODF, nous faisons partie de cette alliance qui promeut ce standard d'utilisation[1]. »*

C'est en relisant nos entretiens que nous nous sommes aperçus qu'il était possible de rechercher et d'acquérir des connaissances de manière très rapide grâce à internet. Suite à

[39] Pour l'expression e-entretien nous sommes redevables à Jean-Claude Sardas.

cela, nous avons commencé à utiliser internet comme un outil permettant de mieux comprendre les propos de nos interlocuteurs. Il convient de noter qu'il est très difficile de partager une discussion avec des spécialistes puisqu'ils font très souvent référence à des formats, des concepts techniques qu'un non spécialiste peut difficilement connaître. Par exemple : *« Dirac is a motion compensated hybrid codec like H264. That is it uses 2D spatial transforms plus motion compensation. It has a more flexible GOP structure than MPEG-2, in which any frame can be a reference frame*[40]*. »* Sans internet, il est très difficile de comprendre de quoi notre interlocuteur parle.

2.2.2. De la technique Delphi à la construction d'une web-based Delphi.

Lorsque nous nous sommes intéressés à la question de l'évaluation de l'innovation dans l'open source, nous nous sommes aperçus qu'il était nécessaire de coupler les techniques quantitatives et qualitatives. Dans la présente section, nous restituons notre démarche de manière séquentielle bien que celle-ci fut structurée au fur et à mesure de notre travail. Il s'agit donc d'une structuration *ex-post* même si notre démarche initiale était beaucoup plus dynamique.

Selon Rowe and Wright, la Technique Delphi fut développée dans les années 40 par une équipe de la RAND Corp. lorsque ces derniers étaient impliqués dans un projet pour l'US Air Force. Cette technique peut être employée lorsqu'il y a un manque de données appropriées et lorsque le jugement humain est nécessaire (Rowe et Wright 1999: 354). Pour Rowe et Wright, il y a quatre conditions pour nommer une technique « *Delphi »* : l'anonymat, l'itération, le retour contrôlé et l'agrégation statistique des réponses. La méthode Delphi nécessite un nombre important d'experts et une itération à plusieurs reprises.

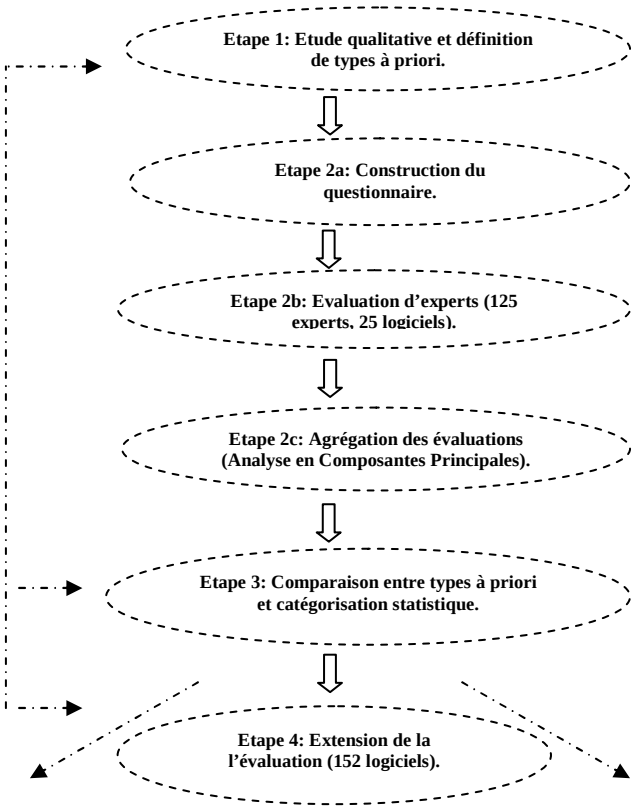
Notre recherche fut menée par le biais d'internet avec le concours d'experts provenant de toute la planète (26 pays sont représentés), c'est la raison pour laquelle nous ne pouvions pas appliquer *l'idéal* décrit par Rowe and Wright. Nous avons plutôt appliqué une variante de la technique Delphi comme cela est décrit par les mêmes auteurs (Rowe et Wright 1999: 354-55).

Notre méthodologie a consisté en quatre phases : (étape 1) la définition qualitative de cinq types d'innovation logiciel basé sur un nombre important d'études de cas (environ 300), des entretiens avec des experts et une revue de littérature pertinente; (étape 2) une évaluation de 25 logiciels libres utilisant ce que nous nommons la « *web-based Delphi »* par 125 experts;

[40] Tim Borer, Dirac Team Leader, BBC Research, entretien en ligne avec l'auteur, mardi 4 avril 2007.

(étape 3) une modélisation conceptuelle de types d'innovation basé sur le jugement moyen d'experts ; (étape 4) un élargissement de la procédure d'évaluation à 152 logiciels libres. Cette méthodologie peut être résumée par le schéma ci-dessous.

Figure 3 : Couplage de techniques au service de l'évaluation de l'innovation.



2.3. Les études de cas : capturer la variété des formes de l'open source.

Dans notre travail, nous avons étudié de nombreuses organisations produisant ou participant à la production de logiciels libres avant de construire notre modélisation des *régimes de l'open source*. Nous présenterons ce matériau empirique en deux sous parties : les cas ayant eu un rôle majeur dans la construction de notre travail [2.3.1.] et les cas qui ont eu un rôle beaucoup moins décisif [2.3.2.].

2.3.1. Les études de cas majeures.

Nous avons concentré notre analyse sur huit études de cas considérés comme suffisamment hétérogènes en nous basant sur la technique de l'échantillonnage théorique (Eisenhardt et Graebner 2007: 27). Ces cas furent analysés en profondeur entre 2006 et 2009. Les données furent collectées à partir d'une trentaine d'entretiens semi-directifs,

d'informations officielles publiées par ces organisations (newsletters, wikis, etc.) mais aussi en utilisant des informations fournies par ces organisations et des données secondaires. Toutefois, les informations secondaires furent systématiquement validées auprès d'acteurs clés de ces organisations (dirigeants et fondateurs notamment). Le tableau ci-dessous synthétise les cas étudiés en profondeur.

Tableau 5 : Etudes de cas majeures

	Cas majeur 1	**Cas majeur 2**	**Cas majeur 3**	**Cas majeur 4**	**Cas majeur 6**	**Cas majeur 7**	**Cas majeur 8**
Noms	Kexi	Freeworks	Mandriva	OpenOffice.org	THALES D3S	OW2 Consortium	Mozilla
	Une communauté développant un système de gestion de base de données relationnelle.	Une association de loi 1901 gérant différentes initiatives autour du concept de l'open source.	Une entreprise gérant le développement d'une distribution Linux.	Une communauté inter-firmes autour d'une suite bureautique.	Une entreprise de hautes-technologies fortement impliquée dans l'open source	Consortium international dédié aux problématiques du middleware.	Une fondation a but non lucratif développant et promouvant les technologies Mozilla.
Nb de membres	10 membres.	10 membres.	140 employés.	726 membres.	70 000 employés.	60 organisations et 1200 individus.	70 membres.
Moyens/ sources	Courriels.	Courriels.	Courriels.	Courriels.	Courriels, présentations, compte-rendu de projets.	Courriels.	Courriels.
Nb entretiens	4	2	4	13	6	4	2
Statut des personnes	Fondateur et membres actifs.	Fondateur et membre actif.	Président, Responsable marketing, Ingénieur et cooker.	Membres du Community Council, lead, co-lead, développeur, contributeurs et utilisateur.	Responsable du Centre de Compétence, Directeur de la Recherche et de l'Innovation, Open Source Architect, Open Source Engineer.	Représentant corporate (Ubikis, Experlog), Membre du board (Thales), responsable local chapter (Thales), Directeur Technique (Bull).	Président et co-fondateur (Mozilla Europe), CEO (Mozilla Messaging).

2.3.2. Les études de cas mineures.

Parallèlement à ces études de cas, nous avons étudié une douzaine d'autres cas à partir d'entretiens semi-directif mais de manière beaucoup moins approfondie. Ces cas eurent un rôle beaucoup moins important dans la construction de notre thèse. Toutefois ils furent mobilisés tout au long de notre travail de façon directe ou indirecte.

Tableau 6 : Etudes de cas mineures.

	Cas mineur 1	Cas mineur 2	Cas mineur 3	Cas mineur 4	Cas mineur 6	Cas mineur 7
Noms	Adonthell	TDR	AFUL	Claroline/Dokeos	Issendis	NeoOffice
	Une communauté développant un jeu en 2D.	Une communauté développant un jeu de rôle en 2D.	Une association de promotion des logiciels libres.	Une communauté et une entreprise développant une plate-forme de collaborative.	Un éditeur de logiciels intégrant la suite OpenOffice.org	Communauté d'utilisateurs-développeurs développant un fork d'OpenOffice.org pour Mac OS X
Nb de membres	5	19	-	-	20	-
Source principale d'information	Données primaires.	Données primaires.	Données primaires.	Données primaires.	Données primaires.	Données primaires.
Nb entretiens	1	1	2	1	1	1
Autres sources	-	-	Courriels.	Courriels.	Courriels.	Courriels.
Statut des personnes	Fondateur.	Fondateur.	Membre du conseil d'administration.	Membre fondateur d'un groupe d'utilisateurs et de la communauté.	Fondateur et directeur commercial.	Membre de la communauté.

Tableau 7 : Etudes de cas mineures (suite).

	Cas mineur 8	Cas mineur 9	Cas mineur 10	Cas mineur 11	Cas mineur 12
Noms	Mekensleep	Wallix	C,mm,n	Apache	Photoshop
	Une société développant des univers persistants basés sur des logiciels libres.	Un éditeur de logiciels libres.	Une communauté impliquée dans un projet d'automobile open source.	Une fondation assurant le développant d'un serveur et d'autres technologies liées à internet.	Logiciel de graphisme.
Nb de membres	-	-	-	-	-
Source principale d'information	Données primaires.	Données primaires.	Données primaires.	Données secondaires.	Données secondaires.
Nb interviews.	1	1	1	1	0
Autres sources	Echanges de courriels.	-	-	Echanges de courriels.	-
Statut des personnes	Directeur Technique.	Directeur des opérations.	Fondateur.	Fondateur.	-

Après avoir présenté les paradoxes ayant émergé suite à la diffusion et au succès des logiciels libres dans l'industrie informatique, nous réaliserons un état des connaissances sur les organisations de l'open source [Chapitre 3]. Cette revue de la littérature nous permettra de mieux présenter l'apport de nos travaux dans la partie suivante [1.].

Nous nous intéresserons en premier lieu aux communautés et plus spécifiquement à la littérature présentant les communautés comme modèle d'innovation distribué [1.1.] et à ceux traitant de la gouvernance et de la division du travail au sein des communautés [1.2.]. Puis, nous nous pencherons sur les travaux portant sur l'implication des firmes dans l'open source [2.] avec une première focalisation sur le rapport entre firme et communautés [2.1.] et un second sur le rôle des firmes dans la fondation de communautés [2.2.].

Chapitre 3. LES DEBATS DE LA LITTERATURE SUR LES FORMES D'ORGANISATIONS DE L'OPEN SOURCE.

1. Les structures des communautés de l'open source.

1.1. Les communautés de l'open source comme modèle d'innovation distribué.

Il y a un ensemble de travaux qui considèrent les communautés open source comme un modèle d'innovation composé d'acteurs indépendants et volontaires utilisant leurs propres ressources pour produire un logiciel libre. Les premiers travaux mettaient particulièrement en avant l'aspect distribué et bénévole des développeurs de logiciels libres (Kogut et Metiu 2000, 2001b, 2001a).

Parmi cette littérature, il y a les travaux de Kogut et Metiu où ils opposent deux modèles d'innovation dans le domaine de la production de logiciels. (1) Le premier modèle est intitulé : *« Optimizing the Cost-Speed Tradeoff » model* (Kogut et Metiu 2000: 10) et consiste à exploiter des avantages de coûts[41] en faisant jouer la concurrence internationale. Plus spécifiquement, les firmes transfèrent les tâches routinières du processus de développement à des *sites offshores.* La notion de site offshore désigne un pays où les coûts de production sont inférieurs au pays d'origine. Ces réductions de coûts peuvent concerner les conditions d'établissement, le coût de la main d'œuvre, etc. D'après Kogut et Metiu, cette stratégie fut mise en évidence en premier lieu par M. Cusumano (Cusumano 1991) après une étude menée sur l'industrie japonaise du logiciel. Dans ses travaux, Cusumano démontre que *« la conception de logiciel est passée de l'art à des tâches routinières manipulant des modules standardisés. »* (Kogut et Metiu 2000: 5).

(2) Le second modèle est nommé : modèle de *« l'E-Innovation »* et désigne un modèle d'innovation où les acteurs sont à la fois bénévoles et disséminés géographiquement. Il s'agit donc d'un mode d'innovation distribué (Kogut et Metiu 2000: 10). Les auteurs ajoutent que *« Linux est une preuve importante de l'existence du modèle de l'E-Innovation. L'innovation distribuée, avec aucune entreprise, aucun contrat, créé un produit hautement réussi. »* et que *« le net a le potentiel de rendre les laboratoires de recherche des entreprises étriqués et archaïques. »* (Kogut et Metiu 2000: 4).

[41] Cette stratégie de sélection des zones de production en fonction des coûts associés n'est pas sans rappeler la théorie des coûts comparatifs de D. Ricardo (1772-1823).

Selon Dahlander et Magnusson, les logiciels libres constituent un exemple de processus d'innovation distribué (Dahlander et Magnusson 2008: 629). De nombreux autres auteurs partagent également cette idée (Dahlander et Wallin 2006; Hicks et Pachamanova 2007; Rossi 2009; Stuermer et al. 2009) pour n'en citer que quelques uns. Ce positionnement a d'importantes limites sur la manière d'aborder la question de l'innovation du fait notamment de la confusion entre produit et processus[42].

D'autres auteurs se sont intéressés à la motivation des développeurs de logiciels libres. Lerner et Tirole ont recherché à comprendre *« Pourquoi des milliers de programmeurs de premier ordre contribuent librement à la constitution d'un bien public ? »* Pour Lerner et Tirole, les développeurs sont motivés par des effets de réputation et de carrière professionnelle (Lerner et Tirole 2000: 2).

Johnson a ensuite défendu l'idée que les logiciels libres étaient des biens publics constitués grâce à des investissements privés (Johnson 2002). La production de logiciels libres présentait donc un paradoxe : l'utilisation de ressources privées pour la création d'un bien mis à disposition de tous. Ce paradoxe a reçu ensuite des réponses complémentaires dans d'autres travaux (Benkler 2002; von Hippel et von Krogh 2003; von Krogh et al. 2003b).

Ainsi, von Hippel et von Krogh ont développé le *« Private-Collective Model »*, un modèle alternatif issu de l'opposition entre deux modèles d'innovation : le modèle de l'investissement privé et le modèle de l'action collective. (1) Dans le modèle de l'investissement privé, le concepteur de l'innovation ne divulguera pas sa démarche puisque toute divulgation de la méthode employée réduit le bénéfice qu'il attend de son innovation. Bien au contraire, l'innovateur se contente de donner le résultat de son travail et non le travail lui permettant d'arriver à son produit. (2) Dans le second modèle, intitulé modèle de l'action collective, la philosophie des innovateurs est diamétralement opposée étant donné que les concepteurs de l'innovation participeront à la constitution d'un bien collectif. En revanche, ils fourniront la méthode et le résultat[43] (von Hippel et von Krogh 2003: 212-13). (3) Entre ces deux modèles d'innovation, von Hippel et von Krogh identifient un troisième modèle qu'ils nomment *« Private-Collective Model »*. *« Dans ce modèle, les participants à des projets de logiciels open source utilisent leur propres ressources pour investir de manière privée dans la création d'un code de logiciel. »* (von Hippel et von Krogh 2003: 213).

[42] Nous reviendrons plus longuement sur ce problème dans la Partie V où nous évaluons l'innovation de logiciels libres conçus par des utilisateurs-développeurs.

[43] Pour mieux comprendre ces deux premiers modèles, nous utiliserons une métaphore largement utilisée par les promoteurs du logiciel libre. Il s'agit de l'exemple de deux cuisiniers qui conçoivent un gâteau : le premier va distribuer, contre une contrepartie ou non, le gâteau qu'il a conçu (modèle 1) ; le second va distribuer, contre une contrepartie ou non, son gâteau et sa recette (modèle 2).

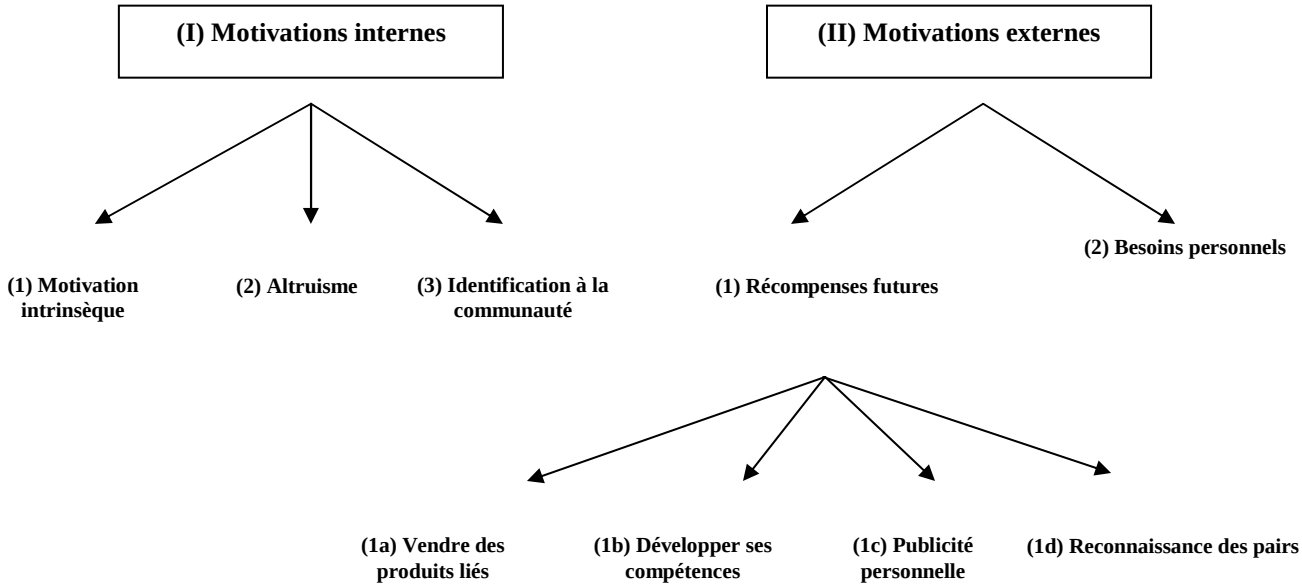
Pour von Hippel et von Krogh, les participants ont un avantage sur les non-participants : il s'agit de l'apprentissage. Les participants obtiennent donc des bénéfices différents des non-participants, c'est pourquoi la question du passager clandestin ne se pose pas dans les communautés open source (von Hippel et von Krogh 2003: 261).

Grâce à l'étude du projet Freenet, von Krogh et al. montrent que parallèlement à la production d'un bien public, le projet développe un pôle de ressources communes conférant des bénéfices pour les développeurs et que ces derniers augmentent en fonction de l'implication (von Krogh et al. 2003b: 3). Plus spécifiquement, les auteurs identifient trois bénéfices : la réputation, le contrôle technologique et l'apprentissage (von Krogh et al. 2003b: 16).

Benkler propose une réponse alternative en expliquant que les individus recherchent des bénéfices indirects comme l'amélioration de la réputation ou la réalisation de prestation de service. De ce fait, le nombre d'utilisateurs augmente le bénéfice attendu de ces logiciels et même s'ils sont passagers clandestins. Benkler, souligne que Weber (2000), appelle ce type de logiciels non pas bien *« non-rival »* où la valeur du bien ne diminue pas lorsqu'il est utilisé mais *« anti-rival »* c'est-à-dire où l'utilisation du bien en augmente la valeur (Benkler 2002: 438).

D'autres études ont recensé l'ensemble des motivations des développeurs de logiciels libres. Hertel et al. expliquent que les développeurs de logiciels libres sont motivés par deux catégories de motivations : d'une part la motivation intrinsèque, c'est-à-dire le plaisir de programmer ; d'autre part les facteurs sociaux conditionnent la compétition entre les développeurs et les entreprises produisant des logiciels fermés (Hertel, Niedner, et Herrmann 2003: 1162). D'après Kogut et Metiu, il existe deux grandes catégories de motivations. (1) D'une part, il y a les motivations intrinsèques principalement caractérisées par l'aspect altruiste des individus provenant d'un côté de la recherche d'une réciprocité et de l'autre de leur identité. (2) D'autre part, les motivations extrinsèques correspondent à la recherche d'une amélioration de la réputation (Kogut et Metiu 2001a: 249). Hars et Ou séparent, quant à eux, les motivations internes et externes pouvant être résumées dans le graphique ci-dessous (Hars et Ou 2002: 27-31).

Figure 4 : Les motivations des développeurs de logiciels libres d'après Hars et Ou (2002).



Le modèle issu des travaux de Hars et Ou (2002) est beaucoup plus large que celui proposé par Kogut et Metiu (2001a), dans la mesure où il intègre aussi bien des éléments liés à la motivation au sens strict du terme mais aussi des éléments concernant l'identité des acteurs.

Dans une étude menée sur le support utilisateurs d'Apache, Lakhani et von Hippel ont distingué quatre types de motivations justifiant que des utilisateurs en aident d'autres : (1) l'apprentissage commun, (2) la promotion de la communauté open source, (3) la réciprocité passée ou future, (4) le sentiment d'obligation (Lakhani et von Hippel 2003: 924).

1.2. La gouvernance et la division du travail dans les communautés open source.

L'un des premiers écrits informant sur la manière dont le développement de logiciels libres s'organisait est le texte de Raymond intitulé *« The Cathedral and the Bazaar »*. Raymond oppose deux styles de développement : d'un côté, le style de la cathédrale caractérisé par un développement où des génies isolés en petits groupes travaillent dans une formidable isolation, sans qu'une version beta ne soit lancée avant que son heure ne soit venue ; d'un autre côté, le style de Linus Torvald ressemblant à un grand *« bazaar »* polyglotte associant différentes approches et manières de développer (Raymond 2000: 2-3).

Dans un premier temps, la littérature considéra que les communautés de l'open source n'étaient régulées par aucune règle et étaient largement auto-organisées à la manière du *« bazaar »*. Le point de départ de cette idée est le travail de Raymond selon qui le processus de développement des logiciels libres est comme une place de marché (*« bazaar »*) où tout le monde peut participer et contribuer dans une atmosphère créative et démocratique (Hertel et al. 2003: 1161).

Rapidement, un grand nombre d'auteurs reprirent cette métaphore et défendirent l'idée que les communautés fonctionnaient sans hiérarchie. Raymond eut une influence très importante sur la littérature puisque son texte figure dans la bibliographie d'un grand nombre d'articles académiques traitant de l'open source (Baldwin et Clark 2006: 1127; Bonaccorsi, Giannangeli, et Rossi 2006: 1098; Cusumano 2004: 27; Dahlander et Magnusson 2005: 493), il est largement considéré comme un classique incontournable dans ce champ de recherche.

Demil et Lecoq ont défendu la thèse que les processus de développement des logiciels libres étaient ni réalisés hiérarchiquement, ni grâce à un mode de coordination marchand, ni en réseau mais à la manière du bazar de Raymond (Demil et Lecocq 2005: 3). Pareillement, d'autres auteurs ont soutenu l'idée que le développement de logiciels libres était réalisé sans coordination hiérarchique entre les acteurs (Elliott et Scacchi 2003: 7) avec toutefois quelques cas particuliers (Garzarelli 2003: 7) comme lorsque le développement est réalisé par une seule entreprise (Henkel 2003b: 7).

D'un autre côté, Bezroukov a contredit Raymond en affirmant que le développement de logiciels libres était comparable à la manière dont la communauté scientifique s'organisait (Hertel et al. 2003: 1161). Kogut et Metiu ont également très tôt souligné qu'une hiérarchie émergeait parmi les développeurs et que celle-ci avait pour mission d'encadrer le développement des logiciels libres (Kogut et Metiu 2000: 2). D'autres universitaires ont par la suite appuyé cette thèse comme Shaikh et Cornford (2003: 128) ou Gauguier (2005: 8).

Il y a un évident paradoxe dans la littérature : certains suggèrent que les logiciels libres sont développés sans hiérarchie et d'autres soutiennent qu'au contraire, il y a bel et bien une hiérarchie. Face à ce débat, nous pensons qu'il est nécessaire de revenir sur ce que les uns et les autres considèrent comme être une hiérarchie. Dans sa thèse, Grassineau a réalisé une analyse critique approfondie de la notion de hiérarchie. Selon lui, la notion de hiérarchie renvoie à au moins trois sens. (1) La hiérarchie peut servir à désigner une forme d'organisation particulière *« caractérisée par un mode d'organisation relativement rigide et des statuts hiérarchisés »*. Nous pouvons raisonnablement considérer cette première définition comme la définition des sciences économiques. En d'autres termes en économie, hiérarchie

désigne le recours à la subordination dans une structure organisationnelle plutôt que le recours au mécanisme de prix (marché). (2) La hiérarchie peut être interprétée comme *« une dotation inégale des pouvoirs entre les acteurs »*. Il s'agit plutôt d'une définition sociologique vue à travers le prisme de la notion de *pouvoir* (Crozier et Friedberg 1977). (3) Pour finir, la notion de hiérarchie peut être considérée sous l'angle *« de l'interaction sociale entre les acteurs d'un même groupe ou de groupes distincts »* (Grassineau 2009: 41-42). Cette définition propose la vision managériale du mot hiérarchie se focalisant sur le rapport de subordination en tant que tel entre un responsable et son subordonné.

Pour répondre à ce paradoxe, les membres des communautés d'utilisateurs-développeurs n'ont ni recours à la hiérarchie (au sens économique[44]) ni au marché pour réaliser leur production (Benkler 2002: 381; Demil et Lecocq 2005: 3). Toutefois bon nombre de communautés fonctionnent de manière hiérarchique[45] (au sens managérial). En effet, nous avons réalisé notre propre étude dans le cadre d'un mémoire de Master Recherche, nous avons constaté empiriquement qu'il y avait bel et bien une hiérarchie entre les membres et que celle-ci était basée sur la contribution (Benkeltoum 2006). De récents travaux ont montré comment se structurait le leadership au sein de la communauté Debian[46] (O'Mahony et Ferraro 2007). O'Mahony et Ferraro ont proposé d'étudier la manière dont un système bureaucratique est intégré dans une communauté. Ils proposent en somme un couplage des formes bureaucratiques et démocratiques.

Les travaux de ces auteurs montrent que le leadership n'est pas seulement basé sur la contribution technique mais aussi sur au moins deux autres éléments. (1) D'une part, l'impact des contributions en termes d'adoption (mesuré par la popularité du paquetage[47] maintenu par le développeur) a un impact significatif sur la propension d'un individu à devenir leader. Plus précisément, le mainteneur a 2,3 fois plus de chances de devenir leader. (2) D'autre part, le fait de rencontrer physiquement les autres développeurs a aussi un impact significatif sur le leadership (O'Mahony et Ferraro 2007: 1098).

O'Mahony, spécialiste des questions de gouvernance dans l'open source grâce à la réalisation d'une thèse sur la manière dont les projets open source sont gérés (O'Mahony 2002), défend l'idée que les notions d'autogestion et d'autogouvernement ne décrivent pas de

[44] Pour ce qui est de la définition de hiérarchie basée sur les pouvoirs, elle est plus difficile à interpréter étant donné que nous n'avons pas analysé les communautés sous l'angle des rapports de pouvoirs.
[45] Nous traiterons plus loin la question de la structuration sociale des régimes de l'open source du point de vue empirique.
[46] Pour de plus amples informations sur l'organisation de la communauté Debian, lire les travaux de N. Auray (2003, 2004, 2006).
[47] Un paquetage est une application intégrée dans une distribution Linux.

manière satisfaisante les modèles de gouvernance dans les projets open source (O'Mahony 2007: 140). O'Mahony suggère de découpler la notion de gouvernance communautaire de la notion d'open source. L'auteur identifie cinq principes de gouvernance communautaire : (1) l'indépendance des structures de direction, (2) le pluralisme des méthodes et techniques, (3) la représentation des membres de la communauté dans les organes de direction, (4) la décentralisation des prises de décisions et (5) la participation autonome des contributeurs (O'Mahony 2007: 148-49).

Sharma et al. soutiennent l'idée que les communautés open source utilisent quatre mécanismes de gouvernance : (1) la gestion des participants (cooptation), (2) des règles et des normes (système de vote), (3) de la surveillance et des sanctions et (4) des effets de réputation (Sharma, Sugumaran, et Rajagopalan 2002: 11).

L'étude du projet Spip montre l'existence de trois types de régulation de l'action collective : (1) un premier centré sur le logiciel, (2) un second autonome reflétant des engagements différenciés et (3) un troisième distribué portant sur la création et la modification de normes partagées (Demazière, Horn, et Zune 2007: 34). Les auteurs ajoutent que *« les projets de grande échelle à la place développent des formes de régulations plus codifiées, stabilisées et prescriptives »* (Demazière et al. 2007: 37). Il est évident que les problématiques de gouvernance ne sont pas les mêmes en fonction de la taille de l'organisation mais cela n'est pas nouveau en théorie des organisations (Beziat 1992: 105). Il semble toutefois qu'une hiérarchie émerge très classiquement lorsque le projet grandit (Maggioni 2002: 14).

2. Les firmes et l'open source.

Au départ, la littérature sur les logiciels libres s'est focalisée sur les communautés d'utilisateurs-développeurs. Néanmoins, la littérature a rapidement mentionné le fait que les firmes avaient un rôle grandissant dans l'open source. Certains universitaires ont particulièrement signalé la nécessité d'étudier l'activité des firmes dans l'open source (Dahlander 2004: 18; Gauguier 2005: 25; Grand, Von Krogh, Leonard, et Swap 2004: 593; West et O'Mahony 2005: 9).

En premier lieu, les recherches se sont focalisées sur l'étude des modèles économiques associés aux logiciels libres (Dahlander 2004; Hecker 2000; 2004b, 2004a, 2005, 2006; Onetti et Capobianco 2005; Pal et Madanmohan 2002; Spiller et Wichmann 2002; Välimäki 2003),

nous présenterons ces travaux dans le chapitre dédié à la valorisation économique du logiciel libre.

Parallèlement à ces études, d'autres auteurs se sont tournés vers les éléments justifiant la participation des firmes dans le développement de logiciels libres. Bonaccorsi et Rossi ont justifié la participation des firmes par trois types de motivations : économiques, technologiques et sociales. Ils mettent en évidence que les motivations économiques sont les plus importantes viennent ensuite les motivations technologiques puis sociales (Bonaccorsi et Rossi 2003b: 23).

De même, Rannou et Ronai suggèrent que les firmes sont impliquées dans le développement de logiciels libres pour deux raisons. (1) D'un côté, les firmes sont intéressées par l'amélioration de la compatibilité de leurs propres logiciels fermés avec d'autres logiciels libres. (2) D'un autre côté, les firmes recherchent à promouvoir des standards pour améliorer l'interopérabilité entre systèmes, réduire les risques d'investissement et donner un meilleur futur à l'innovation (Rannou et Ronai 2003a: 124).

D'autres auteurs se sont intéressés à l'activité des firmes à proprement parler. Grand et al. modélisent les quatre niveaux d'allocation de ressources qu'une entreprise peut avoir dans le but d'exploiter les opportunités offertes par les logiciels libres. (1) Au premier niveau, la firme utilise des logiciels libres, elle n'intervient pas dans le développement des logiciels utilisés. (2) Au second niveau, la firme propose des logiciels libres avec d'autres produits comme atout complémentaire. A ce niveau, les entreprises investissent davantage dans l'open source. (3) Au troisième niveau, la firme sélectionne l'open source comme méthode de production dans le but de profiter du travail des développeurs individuels. (4) Au quatrième niveau, la firme choisit l'open source comme mode de développement. Elle se dote d'un modèle économique compatible avec l'open source (Grand et al. 2004: 594-600).

2.1. Les rapports entre firme et communautés.

Dans un premier temps, les firmes eurent un rôle de financement des communautés. Kogut et Metiu soulignait qu'un nombre important de contributeurs[48] étaient employés par des grandes entreprises de l'industrie informatique (Kogut et Metiu 2000: 28). Plus tardivement, d'autres auteurs signalaient qu'une part importante de développeurs étaient employés par des firmes (Crémer et Gaudeul 2004: 127; Dalle et al. 2005: 407; Ghosh, Glott, Krieger, et Robles 2002; Hars et Ou 2002; Hertel et al. 2003: 1160).

[48] Un contributeur est un individu participant au développement d'un logiciel libre.

Dans un second temps, les firmes participèrent réellement aux communautés existantes. Dahlander et Magnusson ont étudié la dynamique des relations entre entreprises et communautés. Ils classifient les relations existantes entre une firme et une communauté en trois catégories : (1) la symbiose : la relation est profitable pour les deux acteurs ; (2) le commensalisme : la relation profite à un seul acteur sans être préjudiciable pour l'autre ; (3) le parasitisme : la relation procure un bénéfice à un acteur au détriment de l'autre. En revanche, les auteurs insistent toutefois sur le fait que *« les différentes approches ne doivent, néanmoins, pas être vues comme des catégories distinctes, mais plutôt comme les étapes d'un processus continu au regard des bénéfices pour la communauté délibérément recherchés par les firmes de l'open source. »* (Dahlander et Magnusson 2005: 487).

En se basant sur le même matériau empirique traité sous un autre angle, Dahlander et Magnusson ont analysé la manière dont les firmes utilisent les communautés open source dans leurs modèles économiques. Ils identifient trois stratégies : l'accès, l'alignement et l'assimilation (Dahlander et Magnusson 2008). (1) L'accès vise à étendre les ressources de la firme vers l'extérieur en s'appuyant sur deux tactiques : créer de nouvelles communautés et utiliser des communautés existantes. La première technique nécessite plus de ressources mais le contrôle de la firme est plus important tandis que la seconde requiert moins de ressources mais le contrôle est beaucoup plus complexe et indécis. (2) L'alignement consiste à ajuster la stratégie de la firme sur celle de la communauté en employant deux techniques : se servir de licences et influencer l'évolution du logiciel. (3) L'assimilation vise à intégrer et partager les développements avec les communautés par le biais de deux moyens : allouer des ressources pour gérer les contributions communautaires et publier du code sans importance stratégique pour la firme (Dahlander et Magnusson 2008: 637-44). Les travaux de Dahlander et Magnusson prouvent qu'il y a une division implicite du travail entre d'un côté les utilisateurs-développeurs motivés par la création de nouvelles fonctions et de l'autre les développeurs salariés travaillant sur des tâches beaucoup plus routinières comme l'identification et la correction de bugs (Dahlander et Magnusson 2008: 645).

Certaines firmes exploitent le travail des communautés, O'Mahony nomme ce problème le *« détournement »* et considère qu'*« un projet est détourné lorsqu'un vendeur ajoute du code propriétaire au travail de la communauté et tente de le privatiser »* (O'Mahony 2003: 1181). O'Mahony détaille la manière dont les projets open source communautaires protègent leurs travaux contre l'appropriation. Elle identifie sept manières : (1) l'utilisation de licences ; (2) les sanctions légales et normatives ; (3) la création d'une entité juridique ; (4) le transfert des droits à une organisation à but non lucratif ; (5) l'enregistrement de marques et de logos ; (6)

l'assignement de ces droits (marques et logos) à une organisation à but non lucratif ; (7) la protection de la marque du projet (O'Mahony 2003: 1183).

D'autres travaux traitent des relations entre communautés et entreprises mais sous un angle d'analyse différent. O'Mahony décrit l'implication des firmes dans les communautés open source existantes par le biais de ce qu'elle nomme le sponsoring (O'Mahony 2005) ou communauté sponsorisée (West et O'Mahony 2005). West et O'Mahony suggèrent quatre scénarios de développement pour ces communautés. Premièrement, l'entreprise conserve le contrôle sur le développement du logiciel en ouvrant certaines parties seulement du logiciel. Deuxièmement, l'entreprise contrôle fermement le développement du logiciel mais est plus transparente sur le processus de développement. Troisièmement, l'entreprise sponsor délègue le contrôle du développement du logiciel à la communauté. Quatrièmement, l'entreprise et la communauté partagent le contrôle et la responsabilité du développement du logiciel (West et O'Mahony 2005: 4).

Stuermer et al. proposent une extension du *« private-collective model »* (von Hippel et von Krogh 2003) en s'intéressant au cas Nokia Internet Tablet, un produit reposant sur des logiciels à la fois libres et non libres. De manière plus précise, ils se sont intéressés aux composants open source intégrés et à la manière dont ceux-ci sont conçus par des employés de Nokia, des développeurs contractuels et des développeurs bénévoles (Stuermer et al. 2009: 179). Les auteurs ajoutent que Nokia garde un assez grand contrôle au niveau de l'architecture logicielle (Stuermer et al. 2009: 177).

2.2. Les firmes comme initiateurs des communautés.

La littérature s'est focalisée sur deux éléments : d'une part, sur l'étude des communautés d'acteurs indépendants et distribués ; d'autre part, sur l'analyse des rapports entre les firmes et les communautés. Toutefois peu de travaux se sont intéressés aux organisations de l'open source constituées principalement d'entreprises. Seuls deux auteurs étudié ce phénomène.

Tout d'abord, Boiteux oppose deux modes de développement. Premièrement, il identifie le modèle de développement des contributeurs dont la description est globalement très proche des travaux sur les communautés comme réseau d'innovation distribué. C'est pourquoi nous ne nous attarderons pas sur ce mode de développement, nous renvoyons le lecteur intéressé à la section précédente (cf. section 1.1.).

Deuxièmement, Boiteux présente ce qu'il nomme le modèle de développement corporatif. Derrière cette expression, il désigne un modèle de développement réalisé par un concert d'entreprises. D'après Boiteux : *« Le système corporatif permet à plusieurs entreprises qui poursuivent des buts différents, ou identiques, et qui ont besoin des mêmes outils de se réunir autour d'un même projet pour réduire leurs coûts de développement. La transparence du code source, permet à ces entreprises de clairement évaluer la progression du projet, des différents éléments à apporter ou à débattre, mais incite aussi d'autres entreprises à s'intégrer dans les phases de développement. »* Pour lui, nous sommes *« à la genèse d'un nouveau système de production, où le coût de développement se partage entre les différents intervenants, et permet avec des moyens beaucoup moins importants, de proposer des logiciels de grande qualité. »* (Boiteux 2002: partie 5).

Pour Boiteux, il est donc nécessaire de différencier les intérêts des entreprises et ceux des contributeurs car selon lui certaines entreprises profitent de la main d'œuvre *« gratuite »* des développeurs de la communauté réduisant ainsi leurs coûts de production[49].

De son côté, Henkel démontre que les firmes ont littéralement copié le fonctionnement des communautés pour leurs propres développements. Henkel s'est plus précisément intéressé au cas des entreprises commercialisant du matériel embarquant du logiciel libre et Linux en particulier. Henkel met en évidence que ces firmes poursuivent des buts différents et que ces dernières développent des stratégies afin de tirer profit seules des modifications qu'elles ont réalisées sur le code source de Linux. Henkel souligne que dans certains cas, le code d'un appareil n'est révélé qu'après la commercialisation de ce dernier, cette stratégie permet à l'entreprise de garder pendant près d'une année les modifications apportées au code. Lorsqu'elles sont diffusées celles-ci ont perdu leur caractère stratégique. Henkel nomme cette stratégie le *« selective revealing »* (Henkel 2006). Les firmes profitent en fait de la marge accordée par la GPL (General Public License) régissant les conditions de diffusion du noyau Linux. Contrairement à une idée reçue, la GPL n'oblige pas à publier les modifications apportées à un logiciel couvert par cette licence (Henkel 2003a: 14-15).

Dans le cas du Linux embarqué, l'innovation est le résultat *« de contributions complémentaires et hétérogènes »* comme le résultat des musiques sélectionnées sur un jukebox. C'est ce que Henkel nomme le modèle du jukebox (Henkel 2004: 18).

La diffusion du logiciel libre a conduit à une triple révision conceptuelle. La première porte sur l'apparition d'un nouveau type de solidarité compatible avec : une distribution

[49] Le lecteur intéressé par cette dimension peut se référer à la thèse de Benjamin Grassineau (2009).

géographique, une ouverture à d'autres que le groupe initial et une implication des entreprises La seconde concerne le rôle de la production non marchande qui développe des liens complexes avec le marché. La dernière se rapporte à la structure de l'industrie du logiciel puisque le logiciel libre remet en question l'identité de l'éditeur, de l'utilisateur et de l'intégrateur ainsi que les rapports entre ces acteurs.

D'autre part, nous avons présenté nos questions de recherche portant sur trois domaines : l'organisation : où nous nous intéressons au modèle racine à l'origine du premier logiciel libre et aux différentes expansions de celui-ci [Partie II et III] ; la valorisation : où nous étudions l'impact industriel des logiciels libres et la manière dont ils sont valorisés [Partie IV] ; l'innovation : où nous nous focalisons sur le type d'innovations produits dans l'open source ainsi que sur les éléments caractérisant les modèles libres et fermés [Partie V].

La méthodologie de cette thèse a permis la mobilisation de différentes techniques d'investigation. Ainsi, nous avons mobilisé l'approche historienne à la fois pour partir à la recherche des racines des régimes de l'open source mais également pour analyser des études de cas de manière longitudinale. En outre, nous avons mis en exergue le fait qu'internet avait ouvert le champ à de nouvelles techniques de recherche. D'une part, l'entretien par internet introduit une dynamique inédite des savoirs au cours des entretiens : l'apprentissage externe. D'autre part, la création d'un collège d'experts en ligne basé sur la logique et l'esprit de la technique Delphi propose une méthode contingente et innovante pour étudier l'innovation. De manière beaucoup plus classique, nous avons réalisé des études de cas basées sur des entretiens à la fois en ligne et hors ligne.

En ce qui concerne la littérature, nous constatons que celle-ci s'est focalisée sur la caractérisation des communautés constituées d'acteurs indépendants et distribués mais également sur les rapports entre firmes et communautés. Toutefois peu de travaux étudient les organisations constituées majoritairement d'entreprises.

Dans la prochaine partie [Partie II], nous nous intéresserons aux travaux ayant tenté de rattacher le modèle racine à l'origine du premier logiciel libre à des modèles d'organisations existantes en montrant les apports et limites de chacun des cadres proposés. Puis, nous démontrerons que deux notions sont essentielles pour caractériser le modèle de la communauté d'utilisateurs-développeurs à savoir la *solidarité et la distribution*.

PARTIE II. CARACTERISATION ET GENEALOGIE DU MODELE RACINE : LA COMBINAISON INEDITE ENTRE UN SYSTEME DE MANAGEMENT DE LA SOLIDARITE ET UN SYSTEME DE PRODUCTION DISTRIBUE.

La communauté ayant fait l'objet du plus grand nombre d'études dans le domaine du logiciel libre est sans doute la communauté Linux. Le noyau Linux fut initié en 1991 par Linus Torvald. Il proposa une première version de son noyau de système d'exploitation libre en s'inspirant de Minix (une version d'Unix). Rapidement, des développeurs à travers la planète utilisèrent et proposèrent des améliorations pour ce nouveau système. La communauté Linux était donc un modèle de production non-marchand où des acteurs géographiquement dispersés participaient au développement d'un logiciel à la fois libre et gratuit sur leur temps libre sans recevoir (du moins au départ) de compensation financière pour leur travail. Ce modèle fut considéré comme *« un nouveau modèle de production »* (Benkler 2002: 3).

Dans la littérature, différents auteurs se sont appuyés sur plusieurs cadres d'analyse afin de caractériser les communautés d'utilisateurs-développeurs dont la communauté Linux était l'archétype. Dans un premier temps [Chapitre 1], nous vous présenterons les apports et les limites de chacune de ces analyses et notre positionnement. Puis nous montrerons pourquoi il est nécessaire d'utiliser deux notions : *la solidarité* et *la distribution* afin de décrire le modèle de la communauté d'utilisateurs-développeurs de manière pertinente en mettant en évidence ses origines dans l'histoire des systèmes de management de la solidarité [Chapitre 2] et les systèmes de production distribués [Chapitre 3].

Chapitre 1. LES APPORTS ET LIMITES DE LA LITTERATURE EXISTANTE SUR LA CARACTERISATION DES COMMUNAUTES.

Dans le présent chapitre, nous étudierons les travaux tentant de rattacher le modèle de la communauté d'utilisateurs-développeurs à des formes théorisées d'organisations. Ces recherches se partagent en trois cadres dominants : le cadre des communautés de pratiques et épistémiques [1.], le cadre de la *« collective invention »* [2.] et le cadre des réseaux coopératifs [3.].

1. Cadre 1 : Les communautés de pratique et épistémique.

Le premier cadre d'analyse mobilisé dans la littérature sur les communautés d'utilisateurs-développeurs est celui des communautés de pratique. De nombreux chercheurs de ce champ considèrent les communautés d'utilisateurs-développeurs comme des communautés de pratique (Elliott et Scacchi 2003: 21; Madanmohan et Navelkar 2002: 3) à la fois distribuées (Kogut et Metiu 2001a: 258) et non cantonnées à une organisation à la différence des communautés de pratique précédemment étudiées notamment dans les travaux de Brown et Duguid (1991) (Lee et Cole 2003: 635). Sharma et al. suggèrent quant à eux que la construction d'une communauté de pratique est une pré-condition à la constitution d'une communauté (Sharma et al. 2002: 19).

Afin de souligner les limites de ce cadre d'analyse pour caractériser les communautés du logiciel libre, nous commencerons tout d'abord par expliciter le concept de communauté de pratique [1.1]. Puis, nous dégagerons les limites de ce cadre pour la caractérisation des communautés d'utilisateurs-développeurs [1.2]. Et enfin, nous nous interrogerons sur la notion de communauté épistémique [1.3.].

1.1. La notion de communauté de pratique.

Lors d'une intervention, Paul Duguid suggéra une relecture du travail de Julian Orr sur les communautés de pratiques au sein de Xerox. A cette époque, Xerox dominait le marché de la photocopie. Les machines Xerox étaient tellement compliquées que la firme avait du mal à expliquer comment celles-ci fonctionnaient à ses clients. Sur de multiples fonctions les utilisateurs n'en utilisaient que quelques unes. D'autre part, il y a avait un autre problème : les techniciens devaient également être très qualifiés pour réparer ces machines. L'erreur réalisée à l'époque par Xerox est que la firme considéra que les individus suivaient aveuglément les règles. Or en réalité, la différence entre les règles et les pratiques est importante (Duguid 2006; Dumez 2007).

En fait, J. Orr montra qu'il existait une différence entre la manière dont une organisation concevait le travail et la réalité de celui-ci (pratiques) (Brown et Duguid 1991: 41). En étudiant le travail de dépanneurs, Orr découvrit que, si les réparateurs suivaient les modes opératoires proposés par l'organisation, ces derniers perdraient des compétences et seraient incapables de réaliser le travail assigné (Brown et Duguid 1991: 43).

Les travaux de Wenger rapportèrent le même phénomène mais cette fois l'auteur s'est basé sur le cas d'une entreprise d'assurance maladie. Conformément au stéréotype ce type d'organisations est caractérisé par une routine des tâches. Bien au contraire, le cas proposé par Wenger suggère que les agents sont constamment en train d'innover pour faire rentrer les cas qu'ils rencontrent dans les règles établies (Chanal 2000: 3).

D'après Ulhoi, les communautés de pratique peuvent être définies comme : *« de petits groupes de personnes qui ont travaillé ensemble pendant une période temps et qui, à travers de longues discussions, ont développé un sens commun de but et un désir de partager des connaissances et des expériences liées au travail. »* (Ulhoi 2004: 1104). Ces communautés sont composées par des *« acteurs qui partagent des pratiques communes et développent des mécanismes généralement très informels de partage des connaissances »* (de Vaujany 2007: 19 cf. note 21). Une communauté de pratique est donc une organisation informelle auto-organisée visant à combler les manquements des outils et procédures fournis par une organisation pour réaliser un travail.

1.2. Les limites du rapprochement entre communauté de pratiques et communauté d'utilisateurs-développeurs.

Il y a quatre différences fondamentales entre la communauté de pratique et la communauté d'utilisateurs-développeurs révélant clairement les limites de ce rapprochement[50]. (1) Bien que Brown et Duguid considèrent que les communautés de pratique peuvent dépasser les frontières de l'entreprise en incorporant les clients et les fournisseurs, les communautés de pratique sont issues d'une organisation existante tandis que les communautés d'utilisateurs-développeurs sont totalement indépendantes de toute organisation externe[51] (O'Mahony et Ferraro 2007: 1081).

(2) Les communautés de pratique regroupent la plupart du temps des individus ayant un métier identique (voir par exemple le cas des réparateurs de photocopieurs dans Brown et Duguid (1991)) alors que les communautés d'utilisateurs-développeurs peuvent regrouper des

[50] Paul Duguid partage également l'idée que les communautés de l'open source ne sont pas pour la plupart des communautés de pratiques. Nous en profitons pour remercier Paul Duguid pour ses éclaircissements sur la notion de communauté de pratique.

[51] Nous parlons bien évidemment des communautés d'utilisateurs-développeurs pures et non pas des autres types de communautés dans l'open source. Pour une distinction des différents types de communautés se référer à la partie III.

individus ayant des métiers bien différents (par exemple : développeur[52], administrateur réseaux, médecin, consultant, musicien, artiste peintre, graphiste, etc.). De plus, Hatchuel et al. ont souligné que la notion de communauté de pratique était au départ un substitut au métier (Hatchuel, Le Masson, et Weil 2002: 14) n'ayant pas d'équivalent en anglais.

(3) Il n'y a quasiment pas de hiérarchie dans les communautés de pratique (Brown et Duguid 1991: 48) alors que nous avons signalé qu'il existait bel et bien une hiérarchie dans les communautés dans des recherches antérieures (Benkeltoum 2006) ceci fut confirmé par O'Mahony et Ferraro (2007).

(4) Selon Brown et Duguid, les communautés de pratique n'existent que de manière informelle et émergent d'elles-mêmes. Elles ne peuvent être créées, elles sont plutôt détectées (Brown et Duguid 1991: 49) alors que les communautés d'utilisateurs-développeurs peuvent être créées de manière délibérée. Pour Duguid, il n'est pas possible de créer une communauté de pratique, c'est une erreur de penser le contraire. La théorie des communautés de pratiques est une théorie de l'apprentissage avant d'être une théorie des organisations (Duguid 2006).

1.3. Les communautés épistémiques.

D'autres auteurs ajoutent que la communauté Linux est une communauté hybride entre la communauté de pratique et la communauté épistémique. La communauté de pratique est orientée vers la réussite d'une activité et la communauté épistémique est tournée vers la création de nouvelles connaissances (Cohendet et al. 2003: 104-05). Cela signifie que la communauté Linux est à la fois dirigée vers la réussite d'une activité et la production de connaissances.

Les frontières entre communauté de pratique et épistémique sont complexes à appréhender car d'un côté l'apprentissage est le résultat d'une pratique (communauté de pratique) tandis que de l'autre la pratique est sous entendue comme le prétexte de l'apprentissage (communauté épistémique). Il y a donc une forte circularité entre ces deux notions qu'il est difficile de distinguer et d'opérationnaliser comme cadre d'analyse. De plus, Tuomi souligne que la notion de *« communauté »* ne fait pas l'objet d'une description claire et que le concept reste utilisé de manière variable et parfois même contradictoire (Ilkaa Tuomi 2001: 21).

Pour résumer, la communauté d'utilisateurs-développeurs n'est pas une communauté de pratique. En revanche, elle dispose d'un caractère épistémique évident (Cohendet et al. 2003),

[52] Ces exemples sont tirés de fonctions occupées par les personnes interrogées lors de nos recherches empiriques.

c'est pourquoi il est nécessaire d'intégrer la dimension apprentissage dans notre modélisation de la communauté d'utilisateurs-développeurs.

2. Cadre 2 : La « collective invention ».

Le second cadre d'analyse proposé dans la littérature s'appuie sur la notion de *« collective invention »* et repose sur les travaux de Robert Allen (1983). Afin de présenter les apports et limites de cette notion et de son utilisation comme cadre d'analyse du phénomène open source [2.2.], nous vous proposons d'abord de revenir sur le travail initial de R. Allen [2.1].

2.1. Les travaux de R. Allen ou l'émergence de la « collective invention ».

Allen affirmait que les économistes recherchèrent à connaître les innovateurs et leurs motivations. Ces derniers identifièrent trois types d'innovateurs : les centres de recherches à but non lucratif et les agences gouvernementales, les firmes réalisant des activités de recherche et développement, et les inventeurs individuels. Allen ajouta qu'il existait un quatrième type d'institutions produisant des innovations qu'il nommait *« collective invention »* (Allen 1983: 1).

En étudiant l'industrie du fer, Allen découvrit que les avancées en termes de hauteur et de température n'étaient pas rattachables à un acteur en particulier mais à un effort collectif sur une période de 20 ans (Allen 1983: 2). Selon ses analyses, trois éléments rendaient possible le phénomène de la *« collective invention »* : (1) l'amélioration générale était due à une série de petites améliorations incrémentales ; (2) les résultats opérationnels des modifications de fourneaux faisaient l'objet d'une divulgation publique ; (3) les informations générées par l'amélioration des fourneaux existants étaient employés pour construire des fourneaux plus hauts et plus chauds (Allen, 1983 : 5). Selon Allen, trois raisons justifiaient que les firmes libèrent des informations techniques : (1) les propriétaires et dirigeants des firmes avaient des ambitions professionnelles s'améliorant avec la libération d'informations ; (2) le mouvement de personnels dans l'industrie et le recours systématique aux consultants extérieurs rendaient difficile la protection d'informations ; (3) le fait qu'une telle action pouvait être profitable (Allen 1983: 17).

En utilisant le cadre de la théorie économique de l'invention, Allen expliquait le phénomène de la *« collective invention »* comme une réponse effective à la structure et à

l'environnement légal de l'industrie du fer. Allen ajoutait que du point de vue légal ces *« inventions »* n'étaient pas *« nouvelles »* ce qui signifie qu'elles n'étaient pas brevetables. Par conséquent, du fait de la concurrence et de la *« non-appropriabilité »* de l'invention, la rente sociale de l'invention est plus importante que la rente économique attendue (Allen 1983: 2-3).

Nous voyons bien qu'ici Allen introduit une nuance importante : ce qu'il nomme *« invention »* n'en est pas vraiment une. Il s'agit plutôt d'un processus d'amélioration des performances d'un outil de production selon des critères prédéfinis. Comme le souligne Ulhoi, dans l'étude d'Allen, il s'agissait d'innovation incrémentales (Ulhoi 2004: 1098).

2.2. Les communautés open source comme cas de « collective invention ».

Sur la lignée des travaux d'Allen, certains auteurs défendent l'idée que le cas de la création de logiciels libres n'est qu'un cas particulier de *« collective invention »* (Meyer 2003: 2; Nuvolari 2003: 3). Ces auteurs pensent cependant qu'il est nécessaire d'élargir la définition proposée par Allen (1983) et d'intégrer les développeurs indépendants (Meyer 2003: 4; Nuvolari 2003: 3). Ainsi, Meyer définit la *« collective invention »* comme *« un processus sur lequel les améliorations ou les résultats expérimentaux à propos d'un processus ou un outil de production sont régulièrement partagés. »* (Meyer 2003: 4). Dans la définition proposée le partage d'informations se situe au niveau d'un processus ou d'un outil de production et non pas sur un objet produit.

Le phénomène du partage d'information lié à une innovation ne semble pas limité à l'open source (Henkel 2004: 3). Selon la littérature, il existe d'autres cas de *« collective invention »* : Nuvolari détaille un cas de conception partagée de moteur à vapeur en Grande-Bretagne (Nuvolari 2003: 13-23) et Meyer décrit le cas Homebrew, un club de passionnés de micro-informatique qui partageaient leurs expériences (Meyer 2003: 12-14).

D'après Henkel, le phénomène de la *« collective invention »* est comparable à la notion d'*« open innovation »* (Chesbrough 2003). Il ajoute que les firmes décrites par (Allen 1983) sont motivées par un même objectif : l'amélioration des performances des hauts-fourneaux tandis que dans le cas du Linux embarqué, les firmes ont des objectifs divergents (Henkel 2006: 955). Cette limite nous semble généralisable aux communautés composées d'individus du fait de la variété des motivations expliquant leur participation au développement de logiciels libres[53].

[53] A ce sujet se référer au chapitre 3 de cette même partie.

Dans l'ensemble des cas de *« collective invention »* lorsque la technologie entrait dans une phase d'exploitation ces collectifs disparaissaient. Bien que les logiciels libres soient dans une phase d'exploitation cela n'a pas conduit à l'arrêt des projets comme dans les cas de *« collective invention »* (Osterloh et Rota 2007: 163) ces deux éléments montrent les limites du rapprochement entre open source et collective-invention.

3. Cadre 3 : Les réseaux coopératifs.

Le troisième cadre d'analyse repose sur la littérature sur les réseaux[54] coopératifs (Benkler 2002, 2006; Grassineau 2009; Mance 2003). Il s'agit de travaux dont le contenu est le plus satisfaisant pour caractériser les communautés d'utilisateurs-développeurs. C'est le cadre avec lequel nous partageons la plus grande affinité. La notion de *réseau coopératif* fédère différents auteurs. Nous verrons d'abord la littérature portant sur l'économie sociale et solidaire [3.1.]. Puis, nous aborderons les travaux sur la coopération [3.2.].

3.1. L'économie sociale et solidaire.

Tout d'abord, nous nous intéresserons aux travaux de Mance (2003). Tout au long de son ouvrage, Mance remet en question le système capitaliste basé sur la propriété privée des moyens de production, il présente la production et la consommation solidaire comme une solution au problème de la pauvreté et de l'exclusion. Ces travaux apportent une grille d'analyse alternative et originale. Les écrits de Mance portent sur les réseaux de collaboration solidaire qu'il définit comme un *« travail et [une] consommation partagés dont le lien réciproque entre les personnes provient, en tout premier lieu, d'un sens moral de coresponsabilité pour le bien-vivre de tous et de chacun. »* (Mance 2003: 17).

Mance identifie un troisième secteur d'activité correspondant aux activités non lucratives dont le poids est loin d'être négligeable. D'après Lins da Silva, *« aux USA, l'argent mû par les organisations sans but lucratif est supérieur aux produits bruts de tous les pays du monde, à l'exception de sept. »* (Lins da Silva cité dans (Mance 2003: 24)). Mance place ce secteur entre l'Etat et le marché (Mance 2003: 23). Cela montre bien que la dichotomie hiérarchie/marché ne permet pas d'intégrer les formes productives sans but lucratif que

[54] Le lecteur intéressé par la notion de réseaux trouvera des approfondissements dans l'ouvrage collectif suivant : (Callon, Cohendet, Curien, Dalle, Eymard-Duvernay, Foray, et Schenk 1999).

Mance décrit. Nous verrons plus loin qu'il est nécessaire d'avoir un cadre d'analyse intégrant l'ensemble des formes productives marchandes et non marchandes. Toutefois, le couple Etat/marché ne peut pas être adopté car il n'intègre pas les initiatives privées à but non lucratif. Comme le soutient Merges, il existe d'autres formes d'organisations en dehors des gouvernements et des firmes (Merges 2004: 3).

3.2. La communauté d'utilisateurs-développeurs comme réseau coopératif.

Y. Benkler s'est focalisé sur le processus de division du travail pour caractériser les communautés d'utilisateurs-développeurs en mobilisant la classique dichotomie des sciences économiques hiérarchie et marché (Coase 1937). Il explique qu'il existe une troisième forme de production différente des deux autres (Benkler 2002: 381). Ces travaux nomment ce modèle la *« peer-production »* (Benkler et Nissenbaum 2006: 394) et *« fait référence à des systèmes de production qui reposent sur l'action individuelle qui est choisie par l'individu lui-même et décentralisé, plutôt que hiérarchiquement attribuée. »* (Benkler 2006: 62). Benkler souligne que les communautés d'utilisateurs-développeurs sont des *« cas de production par les pairs qui interagissent et collaborent en étant organisé ni [sur un modèle] de marché ni sur un modèle hiérarchique ou managérial. »* (Benkler 2002: 381).

Benkler explique l'émergence du modèle de la *« peer-production »* en se basant à la fois sur la théorie des coûts de transactions (Coase 1937) et la théorie de la propriété (Demsetz 1967). Il soutient que *« lorsque le coût d'organiser une activité sur la base de pairs est inférieur au coût d'utiliser le marché ou une organisation hiérarchique, alors la production par les pairs émerge »* (Benkler 2002: 403).

Ce modèle de production se rapproche de la manière dont les communautés scientifiques fonctionnent depuis des centaines d'années en appliquant le principe de l'évaluation par les pairs. Dans son travail, Benkler met en évidence le fait que les approches basées sur les travaux en sociologie et anthropologie sur le don contre-don et la réciprocité ont l'avantage d'avoir une portée analytique à la fois en ligne et hors ligne. En revanche, il met en exergue le fait qu'elles sont hors du courant dominant des sciences économiques (Benkler 2002: 400). Il ajoute par la suite la portée et le but de son travail : *« Dans cette première étude du phénomène de la production par les pairs, il semble plus important d'établir sa ligne de crédibilité comme un mode de production soutenable et valorisable au travers du cadre*

analytique pertinent le plus utilisé [hiérarchie/marché] plutôt que d'offrir une explication détaillée de son fonctionnement. » (Benkler 2002: 401).

Nous pensons qu'au contraire la dichotomie hiérarchie/marché est un cadre d'analyse réducteur et non adapté. Les travaux de Grassineau ont prouvé que les systèmes de production non-marchands comme les communautés d'utilisateurs-développeurs étaient en dehors du cadre hiérarchie/marché (Grassineau 2009: 26). Par conséquent, nous sommes hors de la sphère de la production marchande et non pas entre hiérarchie et marché comme le prétendent Langlois et Garzarelli (2007: 3). La communauté d'utilisateurs-développeurs est seulement l'exemple le plus typique et contemporain d'autres formes de production en marge de la production marchande avec une logique de fonctionnement différente.

Plus récemment, Benkler a introduit la notion de *« social production »* (Benkler 2006) beaucoup plus large et adaptée pour décrire ces formes de production. D'autre part, Benkler propose d'utiliser la notion de *« commons »* (Hardin 1968) aussi importante pour comprendre ces formes. Il convient de noter que la notion de *« bien commun »* est particulièrement vaste puisqu'elle englobe en réalité plusieurs types de biens[55]. Par exemple, Labatut considère les ressources génétique comme relevant de la catégorie des *« bien communs »* mais aussi l'eau et les ressources de pêche (Labatut 2009: 12).

D'après Benkler, les biens communs peuvent être divisés en quatre types basés sur deux paramètres. Le premier renseigne si le bien est ouvert ou fermé : c'est-à-dire en libre accès à n'importe qui ou réservé à quelques personnes. Le second décrit la manière dont le bien est régulé ou non (Benkler 2006: 61).

L'ouvrage de Benkler[56] est d'une richesse importante en matière d'exemples mais celui-ci ne propose pas véritablement de réflexion théorique sur ce que représentent ces formes de production sociales. Benkler fait seulement référence au fait qu'avec ces nouvelles formes de production il y a une convergence des travaux des anthropologues du don et les économistes (Benkler 2006: 116). La concrétisation de ce rapprochement sera réalisée dans les deux prochains chapitres [Chapitre 2 et 3].

La communauté d'utilisateurs-développeurs est une forme productive particulière puisqu'il s'agit d'un système social non-marchand doté d'une fonction de production. C'est pourquoi nous pensons qu'il faut mobiliser à la fois les sciences économiques, les sciences

[55] Pour une revue des apports et limites des perspectives *« utilitaristes »* et *« naturalistes »* autour de la notion de bien commun voir (Labatut 2009: 14-16).

[56] Les travaux de Benkler ont surtout une portée descriptive et ne proposent pas de passage à la théorie. En d'autres termes, Benkler propose essentiellement ce qu'Albert David nomme des *« faits mis en forme »* (David 2000, 2008).

sociales et les sciences de gestion. C'est une forme d'organisation particulière dont la caractérisation nécessite une analyse profonde des pratiques. Les participants dans ces communautés sont impliqués dans une action collective mais il reste à savoir dans quel type d'action collective ces derniers sont impliqués.

Dalle et al. ont indiqué que ce n'était pas le caractère bénévole du développement des logiciels libres qui était nouveau mais plutôt *« son ampleur et sa vitesse »* et la *« distribution géographique de ses participants »*. Ils ajoutent que les *« modes coopératifs de production de connaissances parmi les membres d'une communauté épistémique distribuée qui n'anticipent pas de rémunération pour leurs efforts n'est pas une innovation sociale récente. »* (Dalle et al. 2005: 397).

Les communautés d'utilisateurs-développeurs sont en fait le résultat de la rencontre historique de deux formes d'organisations : les systèmes de management de la solidarité et les systèmes de production distribués. Ces deux systèmes ont eu des trajectoires de développement relativement distinctes et différentes dans l'histoire des organisations. Dans les deux prochains chapitres nous montrerons que la communauté d'utilisateurs-développeurs est à la fois une organisation de type solidaire [Chapitre 2] et distribuée [Chapitre 3] en faisant la généalogie de chacune de ces formes d'organisations.

Chapitre 2. A LA RECHERCHE DES RACINES DE L'OPEN SOURCE (1) : RETOUR SUR L'HISTOIRE DE L'ACTION COLLECTIVE SOLIDAIRE.

Dans le présent chapitre, nous rechercherons les racines de l'open source par le biais d'une étude historique de l'action collective solidaire. Nous prouverons que la notion de solidarité dans les communautés d'utilisateurs-développeurs est en fait rattachable à une catégorie plus large d'organisations. La littérature sur laquelle repose notre travail est pluridisciplinaire : nous empruntons différentes notions à différents domaines rassemblés du fait de leur forte proximité pratique même si du point de vue théorique, ce sont des champs bien distincts entre lesquels peu de liens furent réalisés. C'est à travers la notion de solidarité que nous réaliserons ce lien [1.]. Pour ce faire, nous détaillerons la notion de solidarité en sciences humaines et sociales [1.1]. Nous définirons ensuite le cadre analytique charité/marché [1.2.]. Puis, en second lieu, nous décrirons les formes productives entre la charité et le marché [2.] avec tout d'abord la souscription se situant entre la charité et le donnant-donnant [2.1] et l'association d'épargne rotative ou l'archétype du donnant-donnant [2.2.]. En troisième lieu, nous détaillerons les formes historiques de la solidarité [3.] avec d'un côté les collectifs à objet de solidarité stable (Utopie, Phalanstère et Corporation) [3.1.] et de l'autre les groupes à objet de solidarité instable (Ghilde, Tontine) [3.2.]. En quatrième lieu, nous spécifierons la notion d'action collective solidaire [4.]. Pour finir, nous distinguerons la communauté d'utilisateurs-développeurs [5.] comme un système de management de la solidarité particulier [5.1.] et au fonctionnement paradoxal [5.2.].

1. De la notion de solidarité à la définition du diptyque charité/marché.

1.1. La notion de solidarité en sciences humaines et sociales.

La notion de solidarité a une origine très ancienne en sciences humaines et sociales. En 1516, Thomas More écrit l'un des premiers traités philosophiques où la solidarité a une place de choix (More 1842, 1987). L'Utopie eut un impact particulièrement grand dans divers domaines des sciences humaines et sociales.

Au XIX^{ème} siècle, différents auteurs placèrent la notion de solidarité au centre de leurs développements. Les travaux de Charles Fourier eurent également un impact important

(Fourier 1822a, 1822b, 1832, 1845a, 1845b). En effet, la théorie du Phalanstère de Charles Fourier fût réalisée comme une critique du *courant dominant* de l'époque largement structuré autour des travaux d'Adam Smith (1776). Fourier soulignait qu'à son époque on écrivait sur la richesse des nations alors qu'en réalité il y avait toujours autant de misère (Fourier 1845a: 35). Il convient de souligner que les travaux de Fourier diffèrent des systèmes collectivistes adoptés en union soviétique. En quelque sorte le système proposé par Fourier se trouve à mis chemin entre le système capitaliste et collectiviste bien qu'ils furent longtemps considérés comme exclusifs et opposés. Dans le tableau ci-dessous, nous mettons en évidence les caractéristiques respectives de chaque système.

Tableau 8 : Les systèmes collectiviste, fouriériste et capitaliste.

	Système collectiviste	**Système fouriériste**	**Système capitaliste**
Principe de fonctionnement.	Chacun travaille pour la production de biens destinés à être consommés en collectif.	Chacun travaille pour la production de biens destinés à être consommés en collectif.	Chacun travaille pour la production de biens de manière séparés ou en groupe qui sont destinés à être vendus sur un marché.
Principe de rétribution.	Rétribution toujours identique quelque soit l'apport de l'individu à la production.	Rétribution proportionnelle au travail, capital et talent.	Rétribution liée à la performance individuelle.

Plusieurs *« disciples »* de Charles Fourier reprirent ses doctrines et créèrent l'Ecole Sociétaire, le lecteur intéressé par ces travaux peut se référer à l'encadré ci-dessous.

Encadré 5 : La naissance de l'Ecole Sociétaire.

Les travaux de Fourier donnèrent naissance à une littérature abondante identifiée ensuite sous le nom d'Ecole Sociétaire (Baudet-Dulary 1832; Chevalier et Considérant 1832; Considérant 1832; Pellarin 1843; Renaud 1845).

L'Ecole sociétaire était constituée de divers individus reprenant la doctrine de Fourier. Par exemple, H. Renaud proposait une synthèse des travaux de Fourier en se focalisant sur la notion de solidarité qu'il considérait comme *« une chose juste et sainte »* (Renaud 1845: 53).

A la mort de Charles Fourier en 1837, Victor Considérant prit la tête de l'Ecole Sociétaire (Vernus 1998: 67) et était opposé aux plusieurs tentatives de Phalanges industrielles qu'il jugeait prématurées notamment à Condé-sur-Vesgre en 1833, à Citeaux en 1841, en Algérie en 1846 (Valette 1980: V) mais aussi au Brésil par Jean Mure sous le nom de Palmitar (Valette 1981: 76). Le Familistère de Godin fondé à Guise en 1859 est aussi né de l'influence de Charles Fourier (Valette 1980: XIII).

En 1887, Ferdinand Tönnies octroyait une place de choix à la notion de solidarité dans *« Gemeinschaft und Gesellschaft »* (Tönnies 1922). L'influence de Tönnies fut très importante sur les travaux d'Emile Durkheim. Toutefois, Durkheim redéfinit les notions de solidarité mécanique et organique avec un sens quasiment opposé à celle de Tönnies (Leif 1922: note 28). Pour une description des notions de « *Gemeinschaft* » *et* « *Gesellschaft* », le lecteur intéressé peut se reporter à l'encadré ci-après

Encadré 6 : *« Gemeinschaft und Gesellschaft »* selon Ferdinand Tönnies.

F. Tönnies distinguait deux types de collectifs : la *Gemeinschaft* et la *Gesellschaft*. (1) La *Gemeinschaft* n'est pas une collection d'individus même si ces derniers sont organisés. Ce terme désigne un groupe d'individus ne pouvant réaliser que des mouvements collectifs. Les membres de la Gemeinschaft sont interdépendants. L'élément liant ces individus est *« un accord silencieux et spontané de plusieurs consciences. »* (Durkheim 1889: 416-22). D'après F. Tönnies, il existe trois formes de Gemeinschaft : (a) la communauté de sang, la plus parfaite des Gemeinschaft ; (b) la communauté de village, constituée de communautés de familles interagissant tellement entre elles qu'elles deviennent interdépendantes ; (c) la communauté des souvenirs et des occupations, rassemblant des individus partageant les mêmes fonctions, les mêmes croyances, les mêmes besoins. Dans toutes ces formes de Gemeinschaft, l'individu travaille pour le bien être de tous. Il n'y a pas d'échange entre les membres de ces communautés ; par voie de fait, il n'y a pas de contrat entre les individus. Ce groupe n'est pas dirigé par des intérêts particuliers mais plutôt par *« les usages, les coutumes, les traditions. »* (Durkheim 1889: 416-22).

Dans (2) la *Gesellschaft,* les individus sont réunis comme dans la Gemeinschaft. En revanche, l'objet de la réunion diffère sensiblement. Ces individus sont en effet uniquement impliqués dans des relations d'intérêts réciproques. L'unité des consciences est troquée contre la pluralité des intérêts. La Gesellschaft est issue de la Gemeinschaft dans la mesure où la Gesellschaft relie plusieurs individus de différentes Gemeinschaft. La grande différence entre ces deux formes est que : la Gemeinschaft est organique, tandis que la Gesellschaft est mécanique. La première est à l'origine de la société primitive, alors que la seconde est à l'origine de la société moderne.

En 1896, les écrits de Léon Bourgeois proposèrent une nouvelle discussion de la notion de solidarité en analysant des formes hétérogènes d'organisations. Pour de plus amples détails sur les écrits de L. Bourgeois nous vous suggérons de lire l'encadré présentant ses travaux.

Encadré 7 : La solidarité selon Léon Bourgeois.

Léon Bourgeois soulignait qu'*« au milieu du siècle, Bastiat et Proudhon ont bien signalé les phénomènes de solidarité « qui se croisent » dans toutes les associations humaines. Mais aucune théorie d'ensemble ne s'est dégagée de ces observations »* et ajoutait que ce mot avait eu peu de succès et que la définition proposée par Littré en 1877 à savoir *« la responsabilité mutuelle qui s'établit entre deux ou plusieurs personnes »* était selon lui une définition *« sans précision et sans portée »* (Bourgeois 1998: 11).

Lors d'un rapport effectué au Congrès d'Education sociale en 1900, L. Bourgeois mettait en évidence le fait qu'*« un certain nombre de sociétés se sont formées volontairement, qui tendent à réaliser entre leurs membres, d'une façon plus ou moins complète, les lois de la solidarité. Plusieurs seront étudiées ici, et l'un des objets du Congrès est précisément de discuter quels sont, parmi les divers types de sociétés actuellement existantes, mutualités, syndicats, coopératives, etc., ceux qui se rapprochent le plus du véritable but. »*. Il ajoutait ensuite *« Depuis l'association purement commerciale qui se préoccupe seulement d'augmenter les profits personnels de ses membres, jusqu'au grandes sociétés coopératives qui organisent l'effort collectif pour réaliser une plus grande somme de justice sociale, il y a toute une hiérarchie de sociétés dans lesquelles l'idée de solidarité reçoit des applications de plus en plus formelles, pour aboutir au type supérieur la réalisation déjà ébauché d'un[e] société coopérative de consommation qui ferait réaliser par ses membres eux-mêmes, groupés d'ailleurs en coopératives de production, l'ensemble des produits consommés, et les assurerait tous contre le cas de déperdition de forces individuelles. »* (Bourgeois 1998: 72). L. Bourgeois s'intéressa à la notion de solidarité mais en ayant une optique très particulière. Nous devons à Léon Bourgeois, le rassemblement de formes organisationnelles hétérogènes analysées sous l'angle de la solidarité.

Nous avons également basé notre réflexion sur le travail d'historiens comme Edwige Praca (2003). L'auteur propose une étude de la solidarité dans l'Hérault entre le XIX^{ème} et le XX^{ème} siècle. Praca travailla plus spécifiquement sur les mutuelles héraultaise. Selon Praca, « *les sociétés d'entraide mutuelle trouvent leur origine à l'époque médiévale dans les confréries religieuses ou les sociétés de compagnonnage à caractère professionnel. »* (Praca 2003: 15). Le phénomène mutualiste est largement alimenté par l'idéologie de l'Ecole sociétaire. En effet, *« Le Suffrage Universel »*, un journal idéologique montpelliérain *« se place explicitement sous le patronage de Proudhon, Louis Blanc, Pierre Leroux et Fourier. »* (Praca 2003: 24).

Du point de vue des théories économiques, Buchanan (1965) étudia plus spécifiquement la propriété des biens. Buchanan montre qu'entra le bien privé et le bien public il y avait une classe de biens intermédiaires. Buchanan introduisit la notion de *« club »* (Buchanan 1965: 1) pour décrire des biens exclusifs. D'un certain point de vue, la théorie des clubs de Buchanan

fait indirectement référence au concept de solidarité puisque selon Buchanan sa théorie des clubs s'applique lorsqu'il y a une possibilité *« d'exclusion »*. Selon lui, la *« non-exclusion »* est une caractéristique des biens publics. Nous verrons à travers le cas des communautés d'utilisateurs-développeurs que les notions d'exclusion et de non exclusion présentent une limite certaine.

Plus récemment, la littérature sur l'économie solidaire redonna à la notion de solidarité un nouveau souffle. Mance décrit donc ce qu'il nomme les réseaux de collaboration solidaire comme un ensemble de cellules[57] de consommation et de production reliées par des liens de consommation et de production (Mance 2003: 67). Ce que Mance décrit c'est en fait un retour à l'autoproduction. Il s'agit donc d'une production en partie autarcique. L'excédent devant être distribué aux autres cellules. Le modèle proposé par Mance est très proche de ce que Fourier proposait dans *« le nouveau monde industriel et sociétaire »* (Fourier 1845a, 1845b) puisque Fourier proposait de vendre l'excédent productif aux autres Phalanges industrielles. Il semble que les réseaux de collaboration solidaire constituent une *redécouverte* du système de Fourier. En outre, il convient de souligner que Jean Mure, membre de l'Ecole Sociétaire, avait réalisé un essai de Phalanstère au Brésil sous le nom de Palmitar (Valette 1981: 76). Il y a peu être un lien entre les réseaux de collaboration solidaire et les idées de l'Ecole Sociétaire en tous cas ces courants ont une grande proximité théorique et philosophique.

Historiquement économistes et sociologues ont eu des objets de recherche différents : d'un côté les économistes étudient les systèmes de production où des opérateurs ont des règles relativement stables régulées notamment par la notion de prix tandis que d'un autre côté les sociologues étudient la construction des règles dans les systèmes sociaux. C'est pourquoi les uns (économistes) se sont focalisés sur les systèmes sociaux de type marchés, hiérarchies, etc. et les autres (sociologues, anthropologues, ethnologues) se sont focalisés sur les systèmes de type famille, groupes ethniques, tribus, etc. Ce que nous constatons actuellement, c'est *une imbrication des structures productives et des systèmes sociaux.* Peu d'auteurs ont analysé à la fois ces deux types d'organisations.

Plus spécifiquement, nous pensons qu'il est nécessaire de combler un vide entre la littérature sur le don d'un côté (Ardener 1964; Axelrod 1992; Mauss 1925) et les sciences économiques et de gestion de l'autre (Buchanan 1965; Coase 1937). Nous prouverons qu'avec ces formes de production nous devons avoir un cadre d'analyse plus large car nous sommes quelque part entre deux extrêmes : la charité et le marché. Ce diptyque est inspiré du

[57] Mance considère les cellules comme les *« unités de base du réseau »* (Mance, 2003 : 67).

« paradigme du don » (Dzimira 2006) proposé par le MAUSS[58] dans la continuation des travaux de Marcel Mauss (1925) mais s'en distingue par certains aspects.

Le MAUSS donne une place de choix au don sans toutefois nier la notion d'intérêt motivant les actions des êtres humains (Dzimira 2006: 2). Notre travail s'inscrit donc dans cette perspective en essayant de réconcilier l'utilitarisme et l'anti-utilitarisme. Pour notre part, il s'agit de montrer qu'entre le don et l'utilitarisme, il y a un pont, un lien possible grâce au diptyque charité/marché. Ce cadre est né au fur et à mesure de nos réflexions sur l'action collective humaine entre l'action charitable et l'action marchande.

A notre connaissance, il n'existe pas de cadre d'analyse intégrant l'ensemble des structures productives allant de l'action marchande à l'action purement bénévole. C'est à travers ce diptyque que nous développerons ce que nous nommons les systèmes de management de la solidarité.

1.2. Définition d'un cadre théorique : le diptyque charité/marché.

La communauté d'utilisateurs-développeurs est une forme de production où des acteurs géographiquement distribués participent à la production d'un bien gratuit et libre[59] d'utilisation sans recevoir de compensation financière pour leur travail. Dans la littérature sur l'open source, la mise à disposition gratuite de logiciels libres est considérée comme un comportement anormal. Toutefois certains auteurs soulignent que l'altruisme est en train de se développer (Dalle et al. 2005: 395-96). En d'autres termes, le comportement utilitariste est estimé comme la normalité. Or, nous verrons qu'il est possible d'interpréter ce type de comportement autrement en mobilisant la littérature sur le don (Dzimira 2006; Mauss 1925). Pour caractériser le fonctionnement de la communauté d'utilisateurs-développeurs, nous réaliserons un détour théorique dans lequel nous définirons un cadre d'analyse *ad hoc*. Contrairement à Benkler (Benkler 2002, 2006) nous ne prendrons pas le diptyque hiérarchie/marché car celui-ci présente de sérieuses limites. Comme nous l'avons précédemment souligné, Grassineau a prouvé que les systèmes de production non-marchands étaient en dehors de ce spectre (Grassineau 2009: 26). Par conséquent, nous élargirons ce

[58] Le MAUSS (Mouvement Anti-Utilitariste dans les Sciences Sociales) *« s'est notamment donné pour objectif d'entreprendre une critique d'un certain utilitarisme, d'une manière de voir les affaires humaines sous le seul angle de l'intérêt individuel calculé. »* (Dzimira, 2006 : 1).

[59] A travers le mot *libre*, nous faisons référence à la notion de liberté telle qu'elle est définie par la Free Software Foundation puisqu'en fait il s'agit d'une *liberté sous contraintes* encadrée par une licence d'utilisation toutefois cette nuance n'a pas une grande importance ici.

cadre afin d'intégrer des formes productives plus hétéroclites. Nous proposons de prendre une nouvelle dichotomie, à savoir, *charité/marché*. Ce diptyque est issu du rassemblement d'un ensemble de travaux en sciences économiques et gestion d'un côté, et d'un groupe de travaux en sciences humaines et sociales de l'autre.

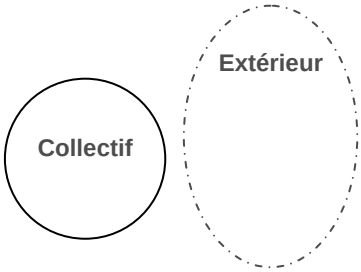
Plus spécifiquement, notre travail s'est largement inspiré des travaux de Marcel Mauss sur le don (Mauss 1925) et du MAUSS. A. Caillé distingue quatre approches du don : (1) les approches économistes du don qui sous-entendent que chaque don est motivé par un intérêt ; (2) les approches niant l'existence du don ; (3) les approches considérant le don comme incomplet ; (4) l'approche maussienne consistant à donner au don différentes fonctions : c'est le paradigme du don (Dzimira 2006: 3-4).

D'après notre compréhension, il semble qu'il y ait plusieurs *classes de dons* ou plutôt *d'actions-réactions* pour utiliser d'autres mots. La notion de *« don »* est tellement marquée par les travaux de Marcel Mauss qu'il est difficile d'octroyer un autre sens à cette notion. C'est pourquoi pour des raisons de clarté nous adopterons un vocabulaire mettant en évidence les distinctions entre les différents types d'actions-réactions. Dans un certain sens, l'action marchande peut être perçue comme un certain type d'action-réaction : une action où la valeur attendue en retour est supposée plus grande par les parties de l'échange. D'un autre côté, notre travail est né également de différentes réflexions sur le rapprochement de phénomènes ayant de fortes similarités en pratique mais ayant fait l'objet de travaux dans des champs théoriques bien distincts.

En s'appuyant sur les travaux de sociologues anti-utilitaristes et hétérodoxes, Grassineau affirme qu'il existe en réalité différents types d'échanges et que la hiérarchie et le marché ne constituent qu'une sorte d'échange (Grassineau 2009: 23). Pour nos propres analyses, nous utiliserons le diptyque charité/marché étant donné que les relations d'échanges entre les individus se situent entre deux extrêmes : l'acte charitable et l'échange marchand. Nous définissons l'acte charitable comme un acte réalisé de manière désintéressée de tout retour matériel. A l'inverse, l'acte marchand est un acte réalisé en vue de recevoir un retour certain dont la valeur est considérée comme supérieure par chacune des parties de l'échange.

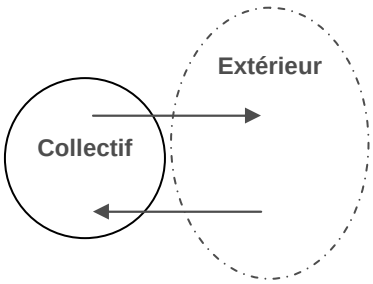
Deuxièmement, notre cadre considérera non pas un individu mais un collectif d'individus et un environnement à l'extérieur de ce collectif. Mauss disait à juste titre *« ce ne sont pas des individus, ce sont des collectivités qui s'obligent mutuellement, échangent et contractent »* (Mauss 1925: 9). Pour être plus clair, nous suggérons le schéma ci-dessous.

Figure 5 : Le couple collectif/extérieur.



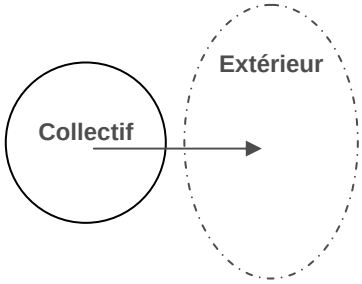
Ensuite, nous envisagerons l'action de ce collectif sur l'extérieur. Considérons que nous sommes dans le cas d'une situation marchande : nous aurons cette fois deux actions symétriques : d'un côté la mise à disposition d'un bien ou d'un service (action du collectif vers l'extérieur) et de l'autre le règlement d'une somme d'argent (action de l'extérieur vers le collectif). Le schéma ci-dessous représente également cette dynamique.

Figure 6 : L'échange marchand.



Envisageons maintenant l'autre extrême (charité), c'est-à-dire une action collective réalisée de manière unilatérale du groupe vers l'extérieur : soit la mise à disposition d'un bien ou d'une somme d'argent à un bénéficiaire strictement situé à l'extérieur du groupe. De manière schématique nous obtenons le graphique ci-dessous.

Figure 7 : L'action charitable.



2. Entre la charité et le marché.

Après avoir posé le cadre *charité/marché*, nous prouverons à travers des cas de structures productives passées et contemporaines qu'il existe des structures de production entre ces deux

extrêmes (charité/marché). Nous analyserons d'une part la souscription [2.1.] et d'autre part la caisse d'épargne rotative [2.1.]. Notre méthode est inspirée de la démarche proposée par Bourgeois (1998) déjà présentée.

2.1. Le cas de la souscription : entre charité et donnant-donnant.

Certaines structures productives ont pour but la réalisation d'une action charitable. Dans cette catégorie d'organisations, nous nous intéresserons à un cas typique : la souscription. Nous en présenterons, l'origine et le système de fonctionnement.

2.1.1. L'origine de la souscription.

La souscription est un système dont la détermination de l'origine est complexe, le lecteur intéressé trouvera quelques informations sur ce thème dans l'encadré ci-dessous.

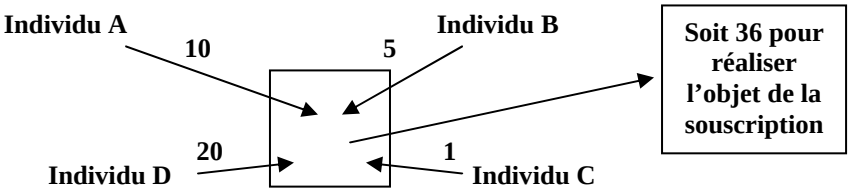
Encadré 8 : L'origine de la souscription.

Le mot *« souscription »* provient du latin *« souscriptio »* qui désignait l'action d'inscrire le nom d'une personne sur une statue ou le fait de signer un document. En 1717, souscription désignait un *« engagement de prendre, moyennant un prix convenu, un ou plusieurs exemplaires d'un livre, d'un ouvrage qui doit être publié »* (CNRS 2004e). Au XVIIIe siècle, le livre était un objet de luxe qui était tiré à un nombre d'exemplaires très faible par rapport à ce que nous connaissons aujourd'hui, la souscription était donc un moyen de mutualiser les coûts. En 1718, le sens de souscription a pris un sens plus large, il signifiait un *« engagement pris de fournir une certaine somme pour quelque entreprise, quelque dépense »* (CNRS 2004e).

2.1.2. Le système de fonctionnement de la souscription

Le processus de la souscription peut être divisé en cinq phases. Premièrement, un individu avait l'idée d'un projet comme la publication d'un livre, la réalisation d'une statue à l'effigie d'un personnage ayant marqué l'histoire, ou la confection d'un présent à l'occasion d'un événement. Deuxièmement, on éditait l'appel à souscription le plus souvent par voie de presse. Troisièmement, on recevait les premières souscriptions pouvant être de simples promesses de dons ou des versements effectifs. Quatrièmement, on publiait le nom des souscripteurs. Et pour finir, on réalisait l'objet de la souscription. La souscription peut-être représentée comme ci-dessous.

Figure 8 : schématisation d'une souscription.



Dans le cas présenté, l'apport des individus n'est pas identique. Il peut être proportionnel au nombre d'unités commandées de l'ouvrage ou bien n'avoir aucune relation de proportionnalité lorsqu'il s'agit d'une action charitable.

D'un certain point de vue, la souscription est une action à caractère purement charitable lorsqu'elle vise à produire un bien public comme par exemple la statue de la liberté. D'un autre côté, lorsque la souscription est simplement un moyen de mutualiser des coûts pour la réalisation d'un ouvrage alors ce n'est plus un collectif ayant action unilatérale sur l'extérieur : c'est un collectif agissant sur lui-même. De manière caricaturale, il existe deux types de souscription : la souscription de type 1 à vocation charitable et la souscription de type 2 correspondant à une action du type donnant-donnant (Axelrod 1992) strictement à l'intérieur d'un groupe. Les schémas ci-après distinguent ces deux types de souscription.

Figure 9 : Souscription de type 1 : charité.

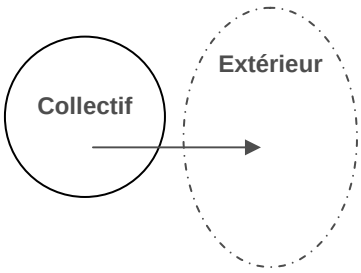
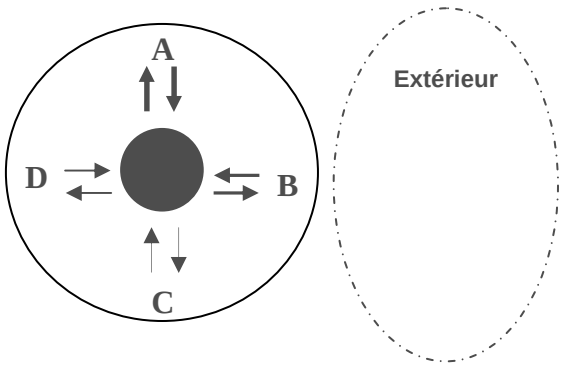


Figure 10 : Souscription de type 2 : mutualisation de coûts.



2.2. Le cas de la « Rotating Credit Association » : l'archétype du donnant-donnant.

2.2.1. L'origine des RCA.

La *« Rotating Credit Association »* (RCA) est une forme alternative de crédit solidaire ayant une origine ancienne dans diverses sociétés. Par exemple, au Japon, la notion de *« Kou »* est attestée depuis le XIIIème siècle. Nous reviendrons plus longuement sur l'origine théorique de la RCA dans l'encadré ci-dessous.

Encadré 9 : L'origine de la *« Rotating Credit Association »*.

L'expression « Rotating Credit Association » (RCA) fut pour la première fois employée par Geertz en 1962. D'après C. Geertz, « le principe de base sur lequel l'association d'épargne rotative est fondée est partout le même : un fond d'une somme forfaitaire composée de contributions fixes de la part de chaque membre de l'association est distribué, à intervalle régulier et comme un tout, à chaque membre chacun son tour » (Geertz cité par Ardener (1964: 201)). Toutefois, pour S. Ardener cette définition était trop restrictive notamment car elle introduisait la notion de « sum » alors qu'il fut démontré par de nombreuses études que les RCA n'étaient pas forcément organisées pour la distribution d'une somme d'argent mais pouvaient également porter sur des biens matériels (i.e.: riz, blé, etc.). De ce fait, Arderner proposa de définir la RCA comme « une association formée par un groupe de participants qui s'accordent pour réaliser des contributions régulières à un fond qui est donné, en totalité ou en partie, à chaque contributeur en rotation. » (Ardener 1964: 201).

Il convient de distinguer la RCA de la tontine bien qu'aujourd'hui le mot tontine soit utilisé pour décrire des formes de RCA. Trois raisons différences justifient cette distinction : (1) dans chacun de ces systèmes, les principes de distribution des rentes sont très différents : dans les RCA, la règle de distribution est fixée avant le commencement du système tandis que dans les tontines la rente est aléatoire ; (2) les RCA peuvent concerner aussi bien de l'argent que d'autres objets (Ardener 1964: 201) alors que les tontines sont uniquement dédiées à l'argent ; (3) dans les tontines l'ensemble du capital est conservé par une institution ou une personne et non pas distribué en suivant un « principe rotatif » (Ardener 1964: 201).

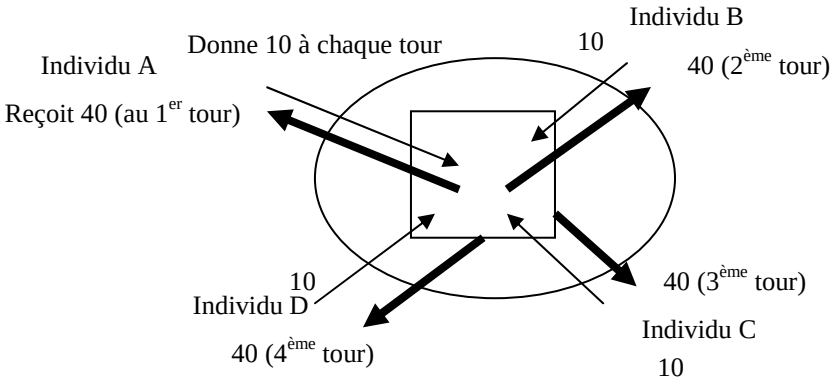
2.2.2. Le fonctionnement des RCA

Les principes de fonctionnement des RCA varient un peu en fonction de la région ou du pays où celles-ci sont organisées, toutefois nous présenterons le principe de base commun à la plupart des RCA. Chaque membre de la RCA s'engage à verser à intervalle de temps régulier une contribution convenue à l'avance par commun accord. Le versement des contributions est

réalisé au cours de réunions, qui d'après Gasse-Hellio sont obligatoires pour tous les participants (Gasse-Hellio 2002: section 2, III B). La non-présence aux réunions peut conduire à l'exclusion du système. L'épargne ainsi récoltée bénéficiera aux participants à tour de rôle. L'épargne dont-il s'agit peut être sous forme d'argent, mais il peut concerner d'autres *« richesses »* (Vanneuville-Zerouki 1999) comme cela a déjà été souligné par Ardener (Ardener 1964: 201). Les RCA sont rarement mixtes et mettent majoritairement en relation des femmes. L'intérêt des RCA est que chaque participant bénéficie de l'épargne de tous, au lieu d'avoir à réaliser seul cette épargne.

Lorsque certains membres reçoivent la cagnotte, ils décident parfois d'arrêter le versement des contributions convenues. Dans ce cas, le conflit entre les participants est réglé socialement : ces individus seront marginalisés, cela signifie qu'ils ne pourront plus participer à une RCA car *la société* n'aura plus confiance en eux. De manière schématique, la RCA fonctionne comme ci-dessous.

Figure 11 : Le fonctionnement d'une association d'épargne rotative.



Dans le cas présenté ci-dessus, chaque individu donne à tour de rôle la même contribution et reçoit une fois sur quatre l'ensemble des contributions. C'est un cas typique de donnant-donnant (Axelrod 1992) comme la souscription de type 2 puisque la somme des contributions est strictement égale à la rétribution obtenue. Ce collectif n'a aucune interaction avec l'extérieur : cela signifie qu'il agit sur lui-même.

Pour résumer l'ensemble des cas étudiés jusqu'ici, nous avons trois types d'actions : la charité (souscription de type 1), le donnant-donnant (souscription de type 2 et association d'épargne rotative) et enfin l'action marchande. Maintenant nous allons nous intéresser à des formes d'actions collectives qui ne sont ni de la charité (action unilatérale de l'intérieur du collectif vers l'extérieur), ni du donnant-donnant (action et réaction symétriques à l'intérieur

d'un collectif) (Axelrod 1992) ou une action marchande (action de d'un collectif vers l'extérieur contre une contrepartie financière).

3. Les formes historiques de la solidarité collective.

Dans la prochaine section, nous traiterons de la question de la solidarité collective. Pour ce faire, nous étudierons deux types de collectifs : un premier groupe où l'objet de la solidarité est relativement stable (Utopie, Phalanstère et Corporation) [3.1] et un deuxième où l'objet de la solidarité est instable (Ghilde et Tontine) [3.2.]. Ces groupes seront analysés à partir de deux éléments : leur origine et leur système de fonctionnement.

3.1. Les collectifs à objet de solidarité stable.

3.1.1. L'Utopie : l'ancêtre théorique des communautés.

A. L'origine de l'Utopie.

L' *« Utopie »* est le nom d'une île imaginaire inventée en 1516 par Thomas More (1478-1535) dans un pamphlet de la société de l'époque. Le lecteur intéressé trouvera une description plus fine de l'origine de l'Utopie dans l'encadré ci-dessous.

Encadré 10 : L'origine de l'Utopie.

Le mot *« Utopie »* est emprunté au latin *« Utopia »* formé par *« ou »* (négation) et *« topos »* (endroit, région) ; littéralement, l'île Utopie désigne *« l'île de nulle part »*.

L'Utopie raconte l'histoire d'une discussion entre T. More, alors en voyage aux Pays-Bas, et un voyageur imaginaire nommé Raphaël Hythlodée. Il lui fit connaître cette île, qu'il décrivait comme *« un vrai modèle de sagesse et de bonne administration »* (Robinet 1733-1777: 662). L'île Utopie avait à l'origine pour nom Abraxa, Utopus, un roi conquérant, lui donna son nom. Utopus a eu l'idée d'humaniser *« une population grossière et sauvage »* (More 1842: 35) en un peuple dépassant l'ensemble des civilisations connues à l'époque.

B. Le système de fonctionnement de l'Utopie.

Le système d'administration de l'île Utopie a fait l'objet d'une description fine : pour de plus amples informations se reporter à l'encadré ci-dessous.

Encadré 11 : L'administration politique de l'Utopie.

L'Utopie était composée de cinquante quatre villes dont la plus grande était Amaurote, la *« ville fantôme »* (Valette, 1981: 13). Cette ville était le lieu où se rassemblaient les représentants de chacune des villes afin de discuter des affaires globales de l'Utopie. Ces représentants étaient au nombre de trois par ville, sélectionnés à la fois pour leur âge et leur expérience, soit cent soixante deux personnes en tout (Robinet 1777: 663).

L'Utopie était administrée par un roi mais il ne disposait que de peu de droits, son rôle se limitait à exécuter et à promouvoir la politique générale de l'île. Les lois étaient quant à elles élaborées par un sénat composé de deux cents membres. Ce sénat se décomposait en deux parties : le sénat élargi comprenant environ deux cents membres et le sénat étroit en comprenant vingt. Les sénateurs de la première catégorie étaient nommés *phylarques* et avaient pour fonction principale et presque unique *« de veiller à ce que personne ne se livre à l'oisiveté et à la paresse, et à ce que tout le monde exerce vaillamment son état »* (More 1842: 40). Quant aux sénateurs de la seconde catégorie, ils étaient nommés *protophylarques* et avaient pour mission de traiter des affaires de l'Utopie avec le roi, en outre ils réglaient les différends entre les utopiens le plus rapidement possible (More 1842: 39).

Bien que l'Utopie n'ait existé que dans la pensée et les écrits de T. More, nous avons choisi de mentionner cette cité. Il est important de noter que l'ouvrage de T. More eut un important impact sur la pensée philosophique et ce très rapidement après sa publication en 1566. De plus, l'Utopie est, semble-t-il, la première forme d'organisation proposant le mariage inattendu, entre la notion de solidarité et celle de production de biens dans un but autre que charitable, mariage que nous ne verrons réapparaître qu'au XIX^{e} siècle avec le Phalanstère de Charles Fourier.

En effet, les utopiens étaient impliqués dans divers travaux au service de la communauté. Par exemple, ils travaillaient à tour de rôle à l'agriculture pendant deux ans, ainsi chaque utopien travaillait pour le bien-être de la communauté entière. Ce système de rotation permettait à chaque utopien d'avoir la garantie qu'il travaillait pour des travailleurs et non des fainéants mais également, de mutualiser les plantations : c'est-à-dire éviter que chacun ne réalisa de son côté ce qui pouvait être mieux fait en groupe (Robinet 1777: 664).

3.1.2. Le Phalanstère : une organisation productive oubliée.

A. L'origine du Phalanstère.

Le Phalanstère fut imaginé par Charles Fourier, il se présentait *« comme inventeur en mécanisme industriel ». « Phalanstère est un mot composé de phalange (corps complet, bataillon organisé) et de la terminaison stère qui, en général, désigne une résidence (sistere, stare)*». Le Phalanstère était donc une *« phalange industrielle »* ou encore une *« entreprise industrielle »* (Chevalier et Considérant 1832: 2-4), un système alliant la notion de solidarité et celle d'industrie.

B. Le système de fonctionnement du Phalanstère

Fourier proposait un procédé d'association dans lequel 1800 personnes produisaient en séries passionnées, sur la base de tâches variées et courtes. Dans le Phalanstère, il introduisit la notion *« d'attraction »* inspirée directement des travaux de Newton sur l'attraction universelle. Fourier expliquait que Newton était passé à côté de l'attraction la plus importante permettant d'obtenir l'harmonie sociale. L' *« attraction passionnée »* avait selon Fourier deux effets : grâce à la passion, le travail devenait attrayant et de ce fait permettait d'améliorer le rendement général. La théorie de la production de Fourier est différente de celle préconisée à l'époque par des théoriciens comme Prony (de Prony 1824) ou Charles Babbage (Aigrain 1980; Babbage 1832).

Dans le Phalanstère, la production réalisée en groupe était également distribuée et consommée en groupe (Chevalier et Considérant 1832: 6). Fourier proposait un comparatif entre production entre l'industrie sociétaire) et l'industrie morcelée.

Tableau 9 : Comparatif entre l'industrie sociétaire et morcelée (Pellarin 1843: 346).

L'industrie sociétaire opère :	**L'industrie morcelée opère :**
Par les plus grandes réunions possibles dans chaque fonction ;	Par les plus petites réunions ou travaux et en ménage ;
Par séances de la plus courte durée et de la plus grande variété ;	Par les séances de la plus longue durée et de la plus grande monotonie ;
Par subdivision la plus détaillée, affectant un groupe de travailleurs à chaque nuance de fonction ;	Par complication la plus grande, affectant à un seul individu toutes les menaces d'une fonction ;
Par l'attraction, le charme.	Par la contrainte, le besoin.

Résultats **De l'industrie sociétaire :**	**De l'industrie morcelée :**
Richesse générale et graduée.	Indigence.
Vérité pratique.	Fourberie.
Liberté effective.	Oppression.
Paix constante.	Guerre.
Températures équilibrées.	Intempéries outrées.
Hygiène préventive.	Maladies provoquées.
Issue ouverte à tous progrès.	Cercle vicieux.
Confiance générale et unité d'action.	Méfiance générale et duplicité de l'action.

Nous citons le Phalanstère comme système de production car plusieurs disciples tentèrent de mettre en place ce système (France, Brésil, Etats-Unis, Algérie). Le Familistère de Guise est aussi un exemple de l'impact de l'Ecole Sociétaire sur les pratiques managériales.

3.1.3. La Corporation : un collectif au service d'un monopole.

A. L'origine de la corporation.

La corporation était un système de défense d'un monopole. La première corporation française concernait les cordonniers de Rouen, attestée par un texte datant d'avant 1135. En ce qui concerne l'origine à proprement parler des corporations, plusieurs thèses furent proposées : l'une d'entre-elles défendait l'idée selon laquelle les corporations n'étaient qu'une application des collegia romaines. D'après Gouron, la thèse de la *« liaison directe et totale »* entre collegia romaine et corporations n'est plus une thèse défendue (Gouron 1958: 38). Plus prudemment, nous dirons que la collegia a été *redécouverte*. Il est évident que du point de vue du concept, la corporation française reste héritière de la collegia romaine.

Pour Gouron, trois éléments expliquent l'apparition des corporations : (1)le regroupement en un même lieu des métiers ; (2) la spécialisation professionnelle et (3) le monopole de production (Gouron 1958: 67-68).

B. Le système de fonctionnement de la corporation.

Une corporation était un corps d'individus (des maîtres) auquel le roi conférait un privilège d'exercer une activité. Les membres d'une corporation jouissaient donc d'un monopole de fabrication et de vente sur un type de produit et sur une zone géographique déterminés (Gourden 1988: 33). Dans les corporations, la solidarité des maîtres portait sur un droit économique commun dont ce système était le garant. Les corporations encadraient les

procédés de fabrication mais ces dernières n'avaient pas vraiment de contrôle sur la production à proprement parler.

Il existait des corporations dans différents métiers : les mouleurs de bois, les orfèvres, les drapiers, etc. Lorsqu'une corporation était établie, l'artisan désirant exercer l'activité encadrée par celle-ci, devait suivre un certain processus appelé *« Maîtrise »*. La première étape de la maîtrise était le compagnonnage : c'est-à-dire l'apprentissage du métier pendant plusieurs années chez un maître. Le passage du statut de compagnon à celui de maître était réalisé grâce à l'élaboration du *« chef d'œuvre »* évalué par d'autres maîtres (Gourden 1988: 29-30).

A travers, ces trois cas historiques impliqués dans une action où la notion de solidarité portait sur un objet relativement stable. Pour l'Utopie et le Phalanstère, la solidarité portait sur la production et la consommation commune de biens. Pour la corporation, l'objet de la solidarité était la défense d'un monopole de production lié à un métier. Maintenant, nous nous intéresserons à des collectifs à objet de solidarité instable : la Ghilde et la Tontine.

3.2. Les collectifs à objet de solidarité instable.

3.2.1. La Ghilde : un système de secours mutuel.

A. L'origine de la ghilde.

La ghilde dispose d'une origine païenne, elle fut cependant adoptée par diverses sociétés malgré l'apparition du christianisme. Pour de plus amples détails sur l'histoire de la ghilde[60] se reporter à l'encadré dédié à ce collectif.

Encadré 12 : L'origine de la Ghilde.

« Ghilde » est le terme latinisé du néerlandais « *gelde »* signifiant *« réunion de fête »*. Dans la France du Moyen Age, la ghilde désignait une association d'intérêts (notamment économiques) entre marchands, artisans ou artistes (CNRS 2004d). D'après F. Augustin, le terme ghilde a deux sens : premièrement, il signifie *« banquet à frais communs »* ; deuxièmement, il désigne une *« association ou confrérie »* (T. Augustin 1840: 269).

La ghilde était un système où les *« convives »* s'aidaient financièrement et matériellement en cas de problèmes. La ghilde était couramment présente dans les pays scandinaves et germaniques. Son origine était païenne puisque chaque ghilde était sous le patronage d'une divinité mais elle fut conservée par les germains en dépit de leur conversion au christianisme (T. Augustin 1840: 271).

[60] Le lecteur souhaitant approfondir ses recherches sur l'histoire de la ghilde pourra se référer à (Spencer 1877).

B. Le système de fonctionnement de la ghilde.

La ghilde peut être considérée comme une confrérie[61]. Dans la ghilde, les membres d'une ghilde étaient nommés *« gueuldons »* et se considéraient comme des frères. La solidarité des gueuldons n'avait pas vraiment de limite en théorie puisque ces derniers devaient prêter secours aux autres membres dans toutes les situations, *« cette promesse de secours et d'appui comprenait tous les périls »* (T. Augustin 1840: 270). Dans les faits, les gueuldons prêtaient serment de fidélité. Si l'un des gueuldons n'honorait pas son obligation celui-ci devait payer une amende et dans certains cas, il pouvait être exclu de la ghilde avec un titre infamant.

La ghilde avait pour objectif de rompre les barrières sociales entre les membres d'une société. Elle visait notamment à rassembler sous une même enseigne *« toute espèce de personnes, depuis le prince et le noble jusqu'au laboureur et à l'artisan libre »* (T. Augustin 1840: 270).

3.2.2. La Tontine : un système d'épargne solidaire.

A. L'origine de la tontine.

La *« tontine »* porte le nom de son inventeur, le banquier italien Lorenzo Tonti (Theuriet 1891: 253), nous procéderons à une description plus fine de l'origine tontine se référer à l'encadré ci-dessous.

Encadré 13 : L'histoire de la Tontine.

Lorenzo Tonti proposa en 1653 cette forme d'association au Cardinal Mazarin. Au XVIII[e] siècle, une tontine était considérée comme *« une sorte de société viagère où ceux qui ont contribué à en former les fonds, se succèdent dans la jouissance des rentes viagères qui la composent, et héritent les uns des autres à mesure qu'il en meurt quelqu'un »* (Robinet 1777: 180). Aujourd'hui, ce terme désigne un *« groupe d'épargnants d'âges différents au sein duquel les parts des associés qui meurent sont réparties entre les survivants, soit qu'ils se partagent le capital accumulé, soit qu'ils bénéficient d'une rente viagère constituée à partir de ce capital »* (CNRS 2004a).

[61] La notion de *« guild »* présente une définition différente dans les pays anglophones. Herbert (1941), décrit l'origine de la ghilde irlandaise comme étant une association de défense de monopole, cette définition fait largement penser à la corporation française. Il y a eu certainement un glissement de sens de l'association de secours mutuel vers la défense d'un monopole toutefois nous prendrons uniquement le sens premier de la ghilde. Pareillement, Merges propose une comparaison de la guilde et des communautés open source. Selon Merges, les guildes et les projets open source ont deux points communs : ils reposent sur des normes partagées et ces normes incluent ce qui ne peut pas être considéré comme la propriété des membres de la communauté (Merges 2004: 20). En revanche, Merges décrit les guildes comme ayant avaient deux fonctions : (1) la protection de droits commerciaux ; (2) l'assurance qualité (Merges 2004: 5).

B. Le système de fonctionnement de la tontine.

Le capital de la tontine était constitué par l'émission d'actions que les participants pouvaient acquérir au comptant ou en plusieurs mensualités. La tontine était divisée en classes correspondant à des tranches d'âges. En retour, chaque tontinier recevait une rente, souvent annuelle, variant en fonction de la classe d'appartenance.

Lorsqu'il y avait des décès dans une classe, les actions des personnes défuntes augmentaient la valeur des actions des vivants. D'une manière simplifiée, la tontine se caractérise comme un système de solidarité entre vivants et morts. D'après Diderot et Alembert, au XVIII^{e} siècle les tontines prenaient un siècle pour s'éteindre (Diderot et d'Alembert 1751-1782: 787a). Si l'ensemble des membres d'une classe venaient à mourir sauf un, alors le dernier tontinier jouissait de l'ensemble de la rente de sa classe[62]. Lorsque tous les tontiniers d'une classe mourraient, la totalité de la rente revenait à l'organisateur de la tontine. Ci-dessous nous vous proposons une schématisation du fonctionnement d'une tontine simplifiée.

A travers les différents collectifs étudiés précédemment, nous avons constaté qu'il existait une forme d'action qui n'était ni de la charité, ni du donnant-donnant au sens strict ou un échange marchand. Dans le tableau ci-dessous, nous vous proposons une synthèse de ces différents collectifs étudiés jusqu'ici.

Tableau 10 : Synthèse des collectifs étudiés.

Nom du collectif	Objet du collectif	Niveau de certitude du retour
Souscription de type 1	Œuvre de bienfaisance	Aucun
Souscription de type 2	Œuvre commune	Très élevé
Rotating Credit Association	Utilisation de la cagnotte	Très élevé
Utopie	Production et consommation partagée	Elevé
Phalanstère	Production et consommation partagée	Elevé
Corporation	Défense d'un monopole	Elevé
Ghilde	Entraide mutuelle	Faible
Tontine	Rente en viager	Faible

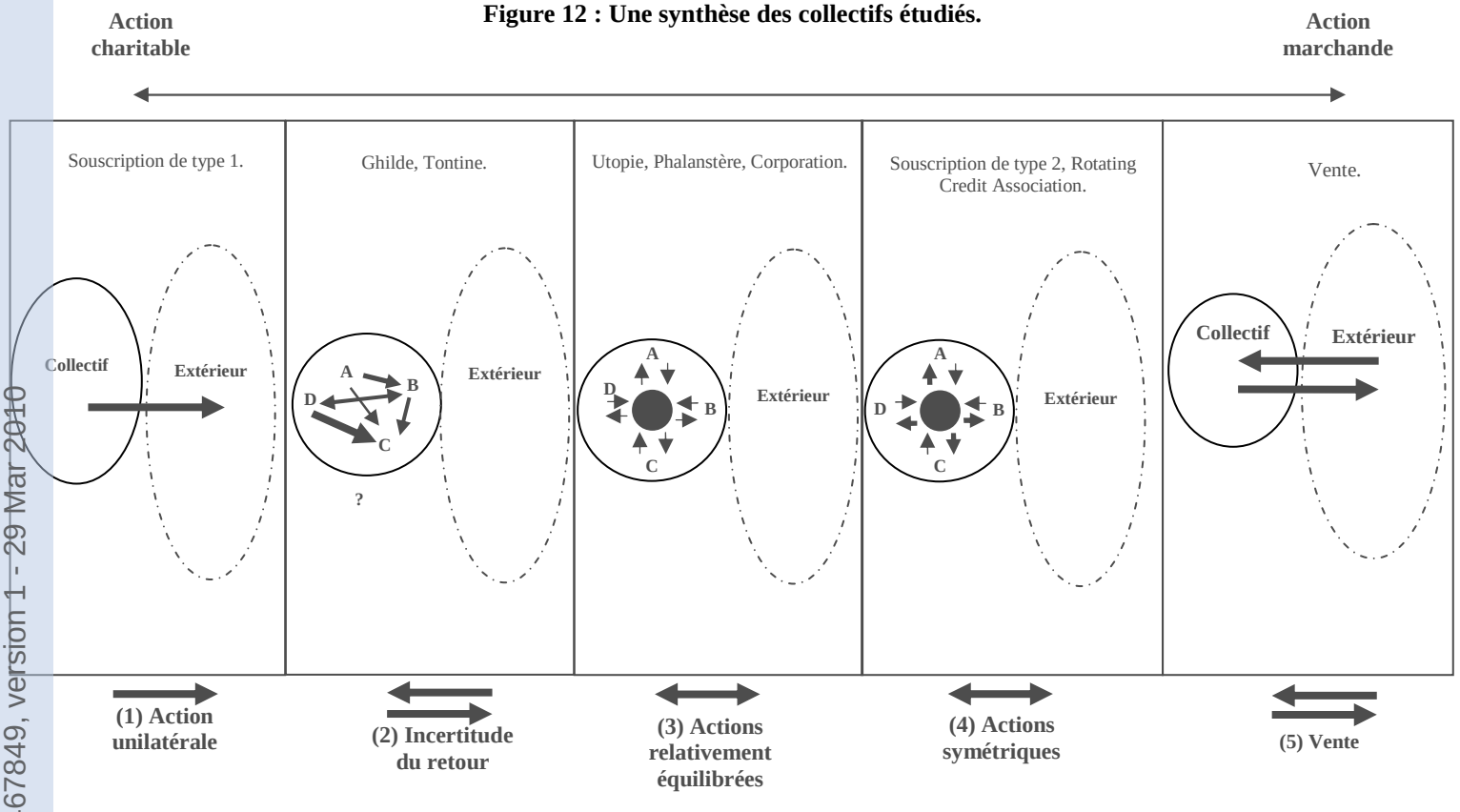
Ces différents collectifs sont classés par degré de certitude. Sur la base de ces études de cas, nous vous proposons de passer à une phase théorique dans lequel nous dégagerons ce que nous considérons être une action collective solidaire.

[62] Cette situation a souvent été critiquée par des auteurs, comme par exemple Stevenson et Osbourne (1984) qui y consacrèrent un ouvrage.

4. Vers la caractérisation d'une action collective solidaire.

Dans le schéma ci-dessous, nous intégrons l'ensemble des collectifs étudiés en utilisant le diptyque *charité*/*marché* choisi comme cadre analytique.

Figure 12 : Une synthèse des collectifs étudiés.



Dans les différents collectifs, nous voyons bien qu'il existe une irrégularité concernant la contribution et la rétribution. Dans le premier cas extrême (1), il n'y a pas de retour : il s'agit d'une action unilatérale. A l'autre extrême (5), nous avons une action dont le retour attendu est considéré comme supérieur par chacun des parties à l'échange. Entre les deux extrêmes, nous avons une action et une réaction parfaitement symétriques (4), puis une action relativement équilibrée (3) et enfin une action dont le retour est incertain (2).

Maintenant, nous dressons les propriétés de ces collectifs de production en mettant en évidence les similarités et différences de chacun. Nous exclurons de notre analyse les collectifs engagés dans une action charitable ou une action marchande.

- Dans (2), (3) et (4), nous avons la coproduction d'un bien pouvant être matériel (Souscription, Association d'épargne rotative, Utopie, Phalanstère), financier (Association d'épargne rotative, Tontine) ou encore immatériel (Corporation, Ghilde). Propriété 1 (P1).
- Dans l'ensemble de ces formes, les acteurs sont volontaires et bénévoles. Propriété 2 (P2).

- Tous ces groupes (2), (3) et (4) sont strictement autarciques c'est-à-dire qu'ils n'ont aucune interaction avec l'extérieur. Propriété 3 (P3).
- Dans (4), il y a une stricte symétrie entre la contribution et la rétribution de chacun. Propriété 4 (P4).
- Dans (3) il y a un équilibre relatif des actions de chacun. Propriété 5 (P5).
- Dans (2), il n'y a pas de symétrie entre la contribution et la rétribution de chacun : il y a une incertitude représentant le fondement du collectif. Propriété 6 (P6).

Afin de résumer les propriétés de ces collectifs, nous suggérons le tableau ci-dessous.

Tableau 11 : Propriétés des collectifs étudiés.

Propriétés	Collectif (2) : (Ghilde, Tontine)	Collectif (3) : (Utopie, Phalanstère, Corporation)	Collectif (4) (Souscription de type 1, RCA)
P1 : Coproduction	X	X	X
P2 : Volontariat et Bénévolat	X	X	X
P3 : Autarcique	X	X	X
P4 : Symétrie C/R[63]			X
P5 : Equilibre relatif C/R		X	
P6 : Incertitude C/R	X		

Nous nous concentrerons sur le collectif dans lequel il y a une incertitude sur le retour (3). A partir des propriétés de cette catégorie de collectifs, que nous définissons ce que nous nommons les Systèmes de Management de la Solidarité (SMS) comme *des systèmes de production où des acteurs coproduisent* (P1) *volontairement et bénévolement* (P2) *un bien, qu'il soit matériel, financier ou immatériel et dont l'usage est strictement réservé à ceux ayant concouru à son élaboration* (P3) *sans qu'il y ait toutefois un équilibre entre la contribution et la rétribution de chacun* (P6). Par conséquent, l'action collective solidaire peut être définie comme une action réalisée dans un groupe mais dont le retour est incertain pour les membres du collectif.

Afin d'avoir une vision claire des différents types d'actions caractérisées au cours des parties précédentes, nous appliquerons notre typologie à l'organisation d'un repas collectif.

[63] C/R = Contribution/Rétribution.

Tableau 12 : Métaphore du repas collectif.

	Repas charitable.	Repas solidariste	Repas d'échange.	Restaurant.
Principe	Le repas est organisé pour d'autres personnes à l'extérieur du groupe.	Chacun apporte ce qu'il veut et peut librement consommer ce qu'il souhaite.	Chacun apporte ce qu'il veut et échange avec les autres en fonction de ce qu'il considère comme étant de même valeur.	Chacun paye ce qu'il a consommé.
Mécanisme de coordination.	Charité.	Solidarité et générosité.	Donnant-donnant.	Marchand.

5. Caractérisation de la communauté d'utilisateurs développeurs.

Nous avons montré qu'il existait différentes formes d'actions collectives entre la charité et le marché. Nominativement les actions collectives décrites sont les suivantes : (1) la charité : l'action unilatérale désintéressée de tout retour ; (2) la solidarité : l'action dont le retour présente une incertitude ; (3) la quasi-solidarité : l'action dont le retour est relativement équilibré en principe ; (4) le donnant-donnant : l'action et la réaction strictement symétrique et enfin (5) la vente : l'action réalisée dans le but de recevoir un retour certain et dont la valeur est considérée comme supérieure.

D'autre part, nous avons vu que la littérature sur l'open source décrivait la communauté d'utilisateurs-développeurs comme une forme productive constituée d'acteurs participant à la production d'un bien libre et gratuit sur leur temps libre. Nous avons également défini le Système de Management de la Solidarité (SMS) comme un système de production où des acteurs coproduisent volontairement et bénévolement un bien, qu'il soit matériel, financier ou immatériel et dont l'usage est strictement réservé à ceux ayant concouru à son élaboration sans qu'il y ait toutefois un équilibre entre la contribution et la rétribution de chacun.

Sur la base de ces éléments, nous prouverons que la communauté d'utilisateurs-développeurs est une forme particulière de système de management de la solidarité [5.1.]. Cela signifie que nous nous concentrerons sur le deuxième type d'action précédemment caractérisé : c'est-à-dire l'action collective solidaire. Puis, nous détaillerons les apports de nos développements théoriques et pratiques vis-à-vis de la littérature [5.2.].

5.1. La communauté d'utilisateurs-développeurs : un cas particulier de Système de Management de la Solidarité.

La communauté d'utilisateurs-développeurs est un système de management de la solidarité ayant quatre particularités détaillées dans l'encadré ci-dessous.

Encadré 14 : Les particularités de la communauté d'utilisateurs-développeurs.

Particularité 1 : Les acteurs de la communauté d'utilisateurs-développeurs participent à la conception d'un logiciel : cela signifie qu'ils sont liés par une solidarité de conception (co-production).

Particularité 2 : Les membres de la communauté d'utilisateurs-développeurs sont souvent intéressés par l'amélioration de leurs connaissances : en d'autres termes ils sont liés par une solidarité d'apprentissage (Cohendet, Créplet, et Dupouët 2003; von Hippel et von Krogh 2003).

Particularité 3 : Les acteurs de la communauté d'utilisateurs-développeurs utilisent l'objet de production mais aussi les acteurs extérieurs utilisant le logiciel : ils sont par conséquent liés par une solidarité d'usage à la fois à l'intérieur et à l'extérieur du collectif.

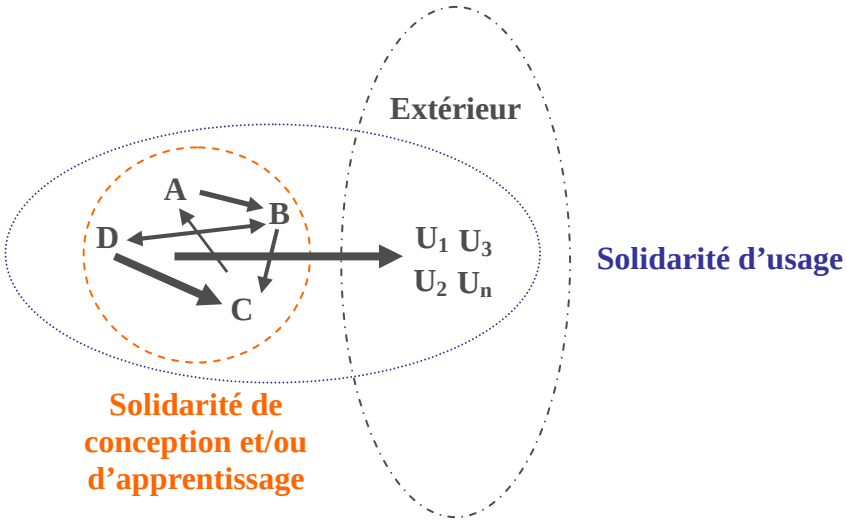
Particularité 4 : L'ensemble des systèmes de management de la solidarité (SMS) ont une frontière opaque sur l'extérieur du collectif. Par exemple : dans la tontine, il fallait faire partie d'une classe pour bénéficier d'une rente en viager ; dans la ghilde, prêter serment pour recevoir l'entraide des autres. La communauté d'utilisateurs-développeurs est le seul SMS ouvert, c'est-à-dire qu'il n'est pas nécessaire de faire partie du système pour profiter du fruit de la production de ce dernier. D'après O'Mahony, *« les contributeurs […] ne souhaitent pas exclure les autres. »* (O'Mahony 2003: 1180). La communauté d'utilisateurs-développeurs a un modèle de *« propriété ouverte »* comme le souligne Grassineau (Grassineau 2009: 16).

La manière dont les communautés développant des logiciels libres utilisent les moyens légaux de propriété intellectuelle leur a permis de découpler : la circulation et la possession du logiciel (O'Mahony 2003: 1195). La communauté d'utilisateurs-développeurs ne fait pas de différence en termes d'usage entre ceux ayant participé à la production du bien commun et ceux n'ayant pas concouru à l'élaboration du bien.

Par conséquent, la communauté d'utilisateurs-développeurs a une action ayant un impact à la fois à l'intérieur et à l'extérieur du groupe. A l'intérieur, il confère des droits de propriété intellectuelle : ce sont les droits d'auteurs puisque l'objet de production est détenu en commun, c'est ce que Benkler nomme les *« commons »* (Benkler et Nissenbaum 2006). A l'extérieur, il donne la possibilité à d'autres d'utiliser gratuitement le logiciel par le biais de la licence d'utilisation. La communauté d'utilisateurs-développeurs est donc le seul SMS

interagissant avec son environnement extérieur en lui conférant des droits d'utilisation de l'objet de production. Le schéma ci-dessous rassemble l'ensemble des caractéristiques de la communauté d'utilisateurs-développeurs.

Figure 13 : Schématisation de la communauté d'utilisateurs-développeurs.



5.2. La communauté d'utilisateurs-développeurs, un SMS au fonctionnement paradoxal : apports à la littérature.

Nous avons vu que *la communauté d'utilisateurs-développeurs* était un Système de Management de la Solidarité (SMS) particulier puisqu'il s'agissait du *seul SMS définissant des règles autres que l'exclusion pour les individus extérieurs au collectif*. Bien que Buchanan soulignait que la *« non-exclusion »* est une caractéristique des biens publics (Buchanan 1965: 13), l*es logiciels libres ne sont pas des biens publics puisqu'ils sont régis d'un côté par le droit d'auteur et d'autre part les conditions de leur diffusion sont encadrées par la contractualisation (licence).* Pour quelle raison s'agit-il du seul SMS de ce type ? Pourquoi les communautés d'utilisateurs-développeurs ne réservent-elles pas l'usage de l'objet produit à ceux qui ont concouru à son élaboration ?

Plusieurs auteurs se sont penchés sur cette question apportant des réponses différentes. (1) Pour von Hippel et von Krogh, les participants ont un avantage sur les non-participants : il s'agit de l'apprentissage (von Hippel et von Krogh 2003: 261). (2) Pour d'autres, les individus recherchent des bénéfices indirects comme l'amélioration de la réputation ou la proposition de services associés (Benkler 2002: 438; Lerner et Tirole 2000: 2). Le nombre d'utilisateurs augmentant le bénéfice attendu et ce même s'ils sont passagers clandestins (Benkler 2002:

438). Autrement dit le problème de la *« Tragedy of the commons »* (Hardin 1968) ne se pose pas pour ce type de biens.

A cette même question, nous ajouterons une explication complémentaire : les communautés d'utilisateurs-développeurs valorisent leur logiciel comme une fonction du nombre d'utilisateurs. Nous citerons pour cela les propos D. Mazzoni, initiateur du logiciel Audacity : *« Cela parait fou, mais parfois, lorsque vous ne gagnez aucun argent, les statistiques à propos du nombre de téléchargement que vous avez c'est vraiment une forme de monnaie*[64]*. »* (Maguire 2007). En d'autres termes, la valeur de l'objet est une fonction du nombre d'utilisateurs : c'est ce que nous nommons *l'effet banquet*.

Nous avons un effet banquet lorsque l'adoption d'un objet détermine la valorisation symbolique de cet objet pour les producteurs. Il s'agit donc d'une situation où des producteurs considèrent que plus il y a d'utilisateurs plus ces derniers valorisent ce bien qu'ils produisent.

[64] Ces propos ont été validés à la fois dans le sens et la portée auprès de Dominic Manzoni.

Chapitre 3. A LA RECHERCHE DES RACINES DE L'OPEN SOURCE (2) : RETOUR SUR LES FORMES DE PRODUCTION DISTRIBUEES.

Dans le chapitre précédent, nous avons signalé que la communauté d'utilisateurs-développeurs était une forme particulière de système de management de la solidarité (SMS), en utilisant l'approche historienne. Dans le présent chapitre, nous prouverons que la communauté d'utilisateurs-développeurs est une forme d'organisation faisant également partie des systèmes de production distribués. Par conséquent, nous procèderons primo à une généalogie des formes de l'action collective distribuée [1.] par le biais de trois cas historiques : la manufacture dispersée [1.1.], l'industrie domestique [1.2.] et la franchise [1.3.]. Secundo, nous présenterons la communauté d'utilisateurs-développeurs comme un système de production distribué [2.] en revenant d'abord sur les caractéristiques communes à l'ensemble des systèmes de productions distribués [2.1.] et plus particulièrement sur les notions de distribution et agglutination [2.2.], et enfin nous dégagerons les éléments ayant permis la conception distribuée de logiciels libres [2.3.].

1. L'histoire des formes d'organisations collectives distribuées.

1.1. La manufacture dispersée de Diderot et Alembert.

D'après Diderot et Alembert, au XVIIIème siècle la littérature sur le commerce était abondante ; celle-ci a tellement été traitée que l'article sur la manufacture est celui *« qu'on lit avec dégoût*[65] *»* (Diderot et d'Alembert 1751-1782). Néanmoins pour ces auteurs, cette matière est loin d'être épuisée.

Selon Diderot et Alembert, le mot manufacture désigne *« un nombre considérable d'ouvriers, réunis dans le même lieu pour faire une sorte d'ouvrage sous les yeux d'un entrepreneur. »* En revanche, le terme manufacture peut aussi désigner une fabrique où le personnel est ni réuni dans un même lieu, ni dans une même ville ; au contraire, elle *« est composée de tous ceux qui s'y emploient, et y concourent en leur particulier, sans y chercher d'autre intérêt que celui que chacun de ces particulier en retire pour soi-même. »* Ainsi dans cet article, Diderot et Alembert distinguent deux types de manufactures : la manufacture

[65] Cette citation et l'ensemble de cette partie est basée sur l'article *« Manufacture »* de Dirderot et d'Alembert (1751-1782).

réunie et la manufacture dispersée. Il existe quatre conditions pour qu'une manufacture réunie soit efficace : (1) les objets de production ne doivent pas varier pas trop souvent ; (2) les sommes dégagées par la manufacture doivent compenser les sommes investies dans la structure ; (3) la manufacture doit être proche des matières premières ; (4) la manufacture nécessite d'être protégée par le gouvernement. Les manufactures réunies concernent : *« les forges, les fonderies, les trifileries, les verreries, les manufactures de porcelaine, de tapisseries »*.

Les manufactures dispersées sont quant à elles adaptées aux ouvrages diversifiés. Elles sont établies avec peu de frais : cela signifie que le coût de revient de fabrication est souvent moindre. Néanmoins, la manufacture dispersée fait l'objet de deux grands défauts : le contrôle du temps des ouvriers et du vol. Les manufactures dispersées concernent *« les fabriques de draps, de serges, de toiles, de velours, petites étoffes de laine et de soie »*.

La grande différence entre les manufactures dispersées et réunies réside dans le type d'ouvrage réalisé : d'un côté dans la manufacture réunie, l'ouvrage nécessite à la fois des fonds, des connaissances techniques et des outils complexes ; d'un autre côté dans la manufacture dispersée, l'ouvrage est relativement simple, l'ouvrier réalisant le travail est souvent assisté par sa femme, ses enfants et domestiques.

1.2. L'industrie domestique.

1.2.1. La définition et la place historique de l'industrie domestique.

Selon G. Renard, la production dispersée a reçu plusieurs appellations dans l'histoire (Renard 1927: 11) : la première est *« système domestique »* et la seconde est *« fabrique collective »* (empruntée à Le Play[66]). Pour G. Renard, l'expression *« fabrique collective »* est maladroite car : *« toute fabrique, que les ouvriers soient rassemblés dans de grands ateliers ou éparpillés dans leurs familles, est, de sa nature, collective. Ce mot, d'ailleurs peut tromper ; il paraît désigner une entreprise dont les différents éléments sont groupés. »* (Renard 1927: 13). Or, le système de production dispersé désigne un système où les ouvriers de production ne sont pas dans un même espace. Ces derniers n'ont pas de contacts entre eux, la coordination entre ces derniers est assurée par un donneur d'ordre qui distribue le travail. De ce fait, G. Renard propose une nouvelle appellation : *« la fabrique dispersée »*. Cette

[66] Frédéric Le Play a notamment dirigé des monographies sur l'industrie domestique sur un tailleur d'habits ou sur une lingère à domicile (Auvray 1862).

expression *« indique que les travailleurs occupés par elle, dépendent d'une seule et même entreprise et en même temps qu'ils sont isolés, disséminés. La fabrique dispersée s'oppose ainsi à la fabrique agglomérée, c'est-à-dire à la manufacture, ou machinofacture, où les travaux s'exécutent dans des ateliers qui réunissent des centaines et des milliers d'ouvriers.* » (Renard 1927: 13-14).

D'après W. J. Ashley, l'industrie a connu quatre phases de développement. (1) Le système de la famille : pendant cette phase, *« l'ouvrage était fait par les membres de la famille pour l'usage de la famille. »*. En d'autres termes, il s'agit d'une autoproduction : la famille produisait ce qu'elle consommait. (2) Le système de la ghilde : dans cette étape, *« l'industrie était exercée par des maîtres pauvres, employant deux ou trois hommes. […] Il y avait un marché, c'est-à-dire qu'il y avait une demande émanant de personnes étrangères à la famille ; mais cette demande était peu importante et relativement stable. »* (3) Le système domestique : lors de cette période, *« il y avait encore de petits maîtres artisans avec des journaliers et des apprentis. […] Mais le maître avait perdu son indépendance économique. »* (4) Le système de la fabrique : pendant cette phase, *« les ouvriers sont assemblés par grandes masses, ordinairement dans de vastes édifices, sous le contrôle immédiat de patrons capitalistes. »* Néanmoins, W. J. Ashley souligne que ces périodes ne devaient pas être considérées comme distinctes dans la mesure où, il existait un grand nombre de stades intermédiaires entre ces phases (Ashley 1900: 259-61).

1.2.2. Le principe de fonctionnement de l'industrie domestique.

D'après Aubert et Gaigné, la dispersion géographique de la production manufacturière est caractérisée par une division du travail basée sur une spécialisation géographique des villes et villages (Aubert et Gaigné 2005: 55)

L'essor de l'industrie domestique est étroitement lié avec l'existence de produits manufacturés nécessitant un faible investissement capitalistique. Les productions dispersées concernaient des tâches ou des opérations incorporant peu de ressources matérielles. De même, pour qu'une production soit réalisée de manière décentralisée, notamment à la campagne, celle-ci ne devait pas nécessiter de qualifications importantes. Dans le cas contraire, c'est-à-dire lorsque des biens requéraient *« du travail qualifié, du capital technique ou une surveillance particulière »* ces derniers étaient produits de manière réunie en ville (Aubert et Gaigné 2005: 57).

1.2.3. Le comparatif entre les systèmes de la fabrique et domestique.

Il y a deux grandes différences entre les systèmes de la fabrique et domestique. Tout d'abord, dans le système de la fabrique, le maître manufacturier disposait d'un grand capital lui permettant de rassembler la main d'œuvre ouvrière en un lieu unique. La surveillance des travailleurs était dans ce cas réalisée par l'entrepreneur, un contremaître ou encore par un autre personnage. A l'inverse, dans le système domestique, la production était réalisée par les maîtres manufacturiers disposant d'un capital peu important. Ils réalisaient leur production avec le concours de leurs proches (femmes et enfants), de journaliers ou encore de leurs domestiques (Ashley 1900: 265-66).

Ensuite, le système de la fabrique permettait à la fois de pousser beaucoup plus loin la division du travail et de contrôler les ouvriers plus efficacement (Ashley 1900: 286). D'autre part, le système domestique était caractérisé par un contrôle du travail en aval : un mode de coordination basé sur une standardisation des résultats (Mintzberg 1982). En outre, la confection à domicile avait une mauvaise image, celle-ci était observable tant du point de vue du type d'individus travaillant chez eux (considérés comme pauvres), que du point de vue de la marchandise confectionnée perçue d'emblée comme étant de mauvaise qualité (Mény 1910: 6-7). En revanche, le système domestique résista aux crises économiques et politiques car celui-ci nécessite peu de frais généraux et se contente de bénéfices réduits. Ces deux atouts furent proposés par A. Audiganne en parlant d'une fabrique de tissus située à Flers dans l'Orne (Audiganne 1860: 99). Nous pensons néanmoins que ces deux éléments sont suffisamment généraux pour s'appliquer à l'ensemble de l'industrie domestique.

1.3. La franchise.

1.3.1. La définition de la franchise.

La franchise est un contrat de collaboration entre deux entreprises indépendantes : d'un côté, le franchiseur concède au franchisé le droit d'utiliser sa marque, son enseigne, son savoir-faire ou tout autre élément pouvant faire l'objet d'une commercialisation ; d'un autre côté, le franchisé paye une redevance et adhère au concept du franchiseur. La franchise est un contrat qui confère au franchisé le droit d'utiliser localement une marque nationale en échange le franchisé doit respecter un niveau de qualité (Merges 2004: 10-11)

La franchise est un contrat assurant des bénéfices à la fois au franchisé et au franchiseur. La franchise offre au franchisé la jouissance d'un nom qu'il n'est plus obligé de construire, il

s'agit donc d'un gain considérable en temps et en sécurité. La franchise permet au franchiseur de développer un réseau de magasins avec moins de fonds qu'un développement en fonds propres et d'être sûr de la motivation de l'entrepreneur franchisé ayant également intérêt à ce que sa franchise soit rentable.

1.3.2. Une typologie de la franchise.

Il existe au moins trois types de contrats de franchise : (1) le contrat de franchise de service confère à un franchisé le droit d'offrir un service sous le nom, l'enseigne ou la marque d'un franchiseur, tout en se conformant aux directives de ce dernier ; (2) le contrat de franchise de production permet à un franchisé de produire lui-même les produits qu'il vend en appliquant le savoir-faire du franchiseur et en utilisant sa marque ; (3) le contrat de franchise de distribution octroie au franchisé le droit de vendre certains produits en utilisant l'enseigne d'un franchiseur (Cour de Justice des Communautés européennes: paragraphe 13)

2. La communauté d'utilisateurs-développeurs : une organisation collective distribuée.

2.1. Caractéristiques communes à l'ensemble des formes d'organisations collectives distribuées.

Suite à l'étude des trois formes d'organisations collectives distribuées sélectionnées, plusieurs remarques peuvent être réalisées. Premièrement, l'organisation distribuée n'est pas nouvelle en soi : la manufacture décrite par Diderot et Alembert en est un excellent exemple.

Deuxièmement, l'organisation distribuée nécessite toujours moins de ressources qu'une organisation centralisée. Ce fait est vérifié à la fois par le comparatif : entre la manufacture dispersée et la manufacture réunie, entre le système de la fabrique et le système domestique mais aussi entre le réseau intégré et la franchise.

Troisièmement, deux caractéristiques sont communes aux systèmes étudiés : *la distribution* du processus de production et *l'agglutination* du résultat de celui-ci. Dans le cas de la manufacture dispersée et de l'industrie domestique, la distribution portait sur des micro-centres de productions dispersés dans les villages et l'agglutination concernait des pièces manufacturées rassemblées par les donneurs d'ordres. En revanche pour la franchise, la distribution et l'agglutination concernaient un objet symbolique (la marque, l'enseigne, le

sigle, etc.). Nous reviendrons sur ces deux notions dans la prochaine section afin de généraliser ces éléments à l'ensemble des systèmes de productions distribués dont la communauté d'utilisateurs-développeurs fait partie.

2.2. Distribution et agglutination : deux notions pour décrire les systèmes de production distribués.

La communauté d'utilisateurs-développeurs, le modèle racine à la base des régimes de l'open source, est également une organisation collective distribuée dans la mesure où cette organisation partage deux grandes caractéristiques communes à l'ensemble des formes d'organisations collectives distribuées : la distribution et l'agglutination.

2.2.1. La distribution

L'action de distribuer peut être définie comme *« donner aux membres d'un groupe où à des personnes, une partie d'une chose ou des choses de même nature faisant partie d'un ensemble »* (CNRS 2004c). Dans le cas de la communauté d'utilisateurs-développeurs, la distribution se trouve au niveau de la disposition des acteurs dans l'espace même si d'après certains praticiens : *« l'open source peut exister sans développement distribué également. On peut avoir une équipe, une société ou un seul individu qui travaille tout seul et qui poste sur internet. Pour moi cette notion de distribution elle n'est pas du tout liée à l'open source en tant que tel*[67]*. »* Et comme le souligne Fuggetta, *« le développement distribué n'est pas la propriété exclusive du logiciel open source. »* (Fuggetta 2003: 83).

2.2.2. L'agglutination

L'action d'agglutiner peut se définir comme *« coller ou se coller fortement de manière à former une masse compacte ou un tout cohérent »* (CNRS 2004b). L'agglutination, pour sa part, concerne les caractéristiques de l'objet distribué. Dans le cas du code informatique, l'agglutination est forte et la modification de l'objet ne le détruit pas contrairement à un article manufacturé par exemple. Cette propriété permet au logiciel d'être en conception constante malgré son utilisation par le client final. En outre, et cela est une propriété propre au

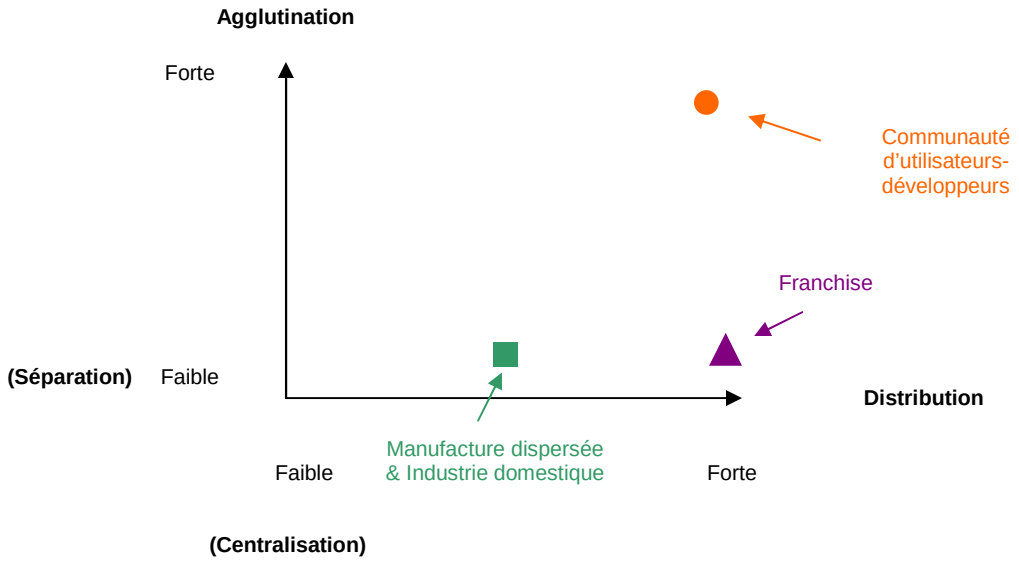
[67] Eric-Marc Mahé, (Technology Advisor, Sun Microsystems France), entretien en face à face avec l'auteur, lundi 15 janvier 2007.

code informatique des logiciels libres : ils peuvent être modifiés avec un impact réduit sur leur structure grâce à leur modularité (MacCormack, Rusnak, et Baldwin 2006: 1015-16) réduisant également les barrières pour contribuer (von Krogh, Spaeth, et Lakhani 2003a: 1218). A l'inverse, dans le cas d'un processus de conception distribué d'une automobile, les acteurs doivent être en constante interaction pour gérer les jeux de pièces (Sardas 2006), nous avons la même situation dans l'aéronautique. Grâce à cette classification, les acteurs peuvent réaliser seuls un travail précis et le rendre rapidement disponible aux autres.

2.2.3. Mise en perspective des systèmes de production distribués.

Pour situer la conception distribuée de logiciels libres par rapport aux différentes formes d'organisations collectives distribuées étudiées, nous suggérons le graphique ci-dessous. Ce graphique reprend les deux notions de distribution et d'agglutination.

Figure 14 : Comparatif des systèmes de production distribués.



2.3. Regard sur les éléments ayant permis l'apparition de la conception distribuée de logiciels par des communautés d'utilisateurs-développeurs.

La communauté d'utilisateurs-développeurs est une organisation où des acteurs sont géographiquement dispersés, ce système de production a été rendu possible grâce à l'apparition conjointe de 4 éléments : un objet virtuel particulier (le code informatique) [2.3.1], le partage de langages communs à l'ensemble de l'industrie informatique (les

langages de programmation) [2.3.2], une infrastructure standardisée pour l'exécution des programmes (le micro-ordinateur) [2.3.3] et enfin un moyen de communication libre (l'internet) [2.3.4].

2.3.1. Un objet virtuel particulier: le code informatique.

A. Le monde des objets virtuels ou symboliques.

Aujourd'hui, nous connaissons au moins trois types d'objets virtuels[68] : la monnaie, la langue et les textes. Le code informatique est une quatrième forme d'objet virtuel.

→ La monnaie.

Pour nous, la monnaie est un objet virtuel depuis que celle-ci n'a plus de valeur intrinsèque. Autrefois, la monnaie avait une valeur propre car elle était souvent conçue à partir d'un métal précieux. Avec le temps, la monnaie s'est détachée de son existence physique avec l'apparition de la lettre de change au XII^{ème} siècle : la valeur de celle-ci dépendait de son émetteur ou encore du billet de banque anonyme au XVII^{ème} siècle.

De nos jours, les flux monétaires sont entièrement informatisés par le biais de jeux d'écritures comptables. La monnaie a perdu sa valeur intrinsèque depuis qu'elle est non convertible en or. Ce phénomène est récent dans l'histoire de l'homme : il faut attendre les accords de Bretton Woods de 1944 pour que toutes les monnaies (excepté le dollar) ne soient plus convertibles en or. Le dollar n'est plus convertible en or à partir de 1971 seulement. Par conséquent, la monnaie a la valeur que les agents économiques lui donnent s'ils n'ont plus confiance en celle-ci, elle peut devenir sans valeur. Lorsque les agents économiques n'ont plus confiance en la monnaie, ces derniers ont recours aux formes alternatives d'échange.

→ La langue.

Il existe trois types de langues : les langues orales (l'anglais, le français, l'espagnol, etc.), les langues des signes et des symboles (le langage des sourds et muets, le code de la route, etc.) et la langue des objets (le braille). Toutes ces langues ont un point en commun : elles n'ont qu'une valeur symbolique. Les langues se bâtissent uniquement sur des conventions tacites passées entre les acteurs de celle-ci. C'est sans doute pour cette raison qu'elles évoluent dans le temps, au gré des interactions.

[68] Il en existe certainement d'autres mais pour les besoins de notre analyse nous n'en avons considéré que quelques-uns.

→ **Les textes.**

Les textes sont également des objets virtuels dans la mesure où si l'on prend les caractères un à un, ces derniers n'ont pas de valeur.

B. Le code informatique : une quatrième forme d'objet virtuel.

Selon nos analyses, le code informatique est un objet virtuel dans la mesure où son existence physique n'a pas de valeur intrinsèque. Un code informatique a de la valeur s'il est associé à des applications ou utilisations. La grande différence entre un code informatique et un texte traditionnel, est que l'ajout ou le retrait de caractères, de termes ou mêmes d'expressions ne conduisent pas à l'aliénation de l'objet. En outre, le code informatique a deux propriétés. Premièrement, du point de vue de l'information c'est un code répliquant. Cela signifie que la copie est strictement identique à l'original. Ce qui n'est pas le cas de la photocopie d'un texte par exemple. Deuxièmement, et cette caractéristique lui est propre, le code informatique se réplique à un coût nul ou quasi-nul.

2.3.2. Des langages communs.

La communication entre des acteurs nécessite un langage partagé. Dans le domaine du logiciel libre, les acteurs produisant conjointement un logiciel devaient avoir une langue identique pour comprendre les instructions écrites par d'autres. C'est ici que les langages de programmation ont joué un rôle très important. Historiquement, la dichotomie entre logiciel libre et logiciel fermé est contingente de deux phénomènes : la standardisation des postes informatiques et l'évolution des langages de programmation.

A. L'impact de la standardisation des postes informatiques.

Dans les années 60, il n'était pas utile de cacher le code source des programmes développés : les programmes étaient développés sur mesure, cela signifie que le programme dans sa globalité était uniquement utilisable pour l'environnement spécifique pour lequel il avait été conçu. Des modifications importantes devaient être apportées pour qu'il soit utilisé sur un autre système. En revanche, l'émergence de standards de fait, notamment due à la suprématie du PC (Personal Computer) d'IBM et de Windows de Microsoft, a considérablement amoindri cet argument.

B. Le rôle de l'évolution des langages de programmation.

La distinction entre logiciel libre et fermé est datée, elle est contingente de l'évolution des langages de programmation utilisés. Nous sommes passés de la programmation en langage machine, à l'assembleur, puis au langage B, C, et aujourd'hui au langage orienté objet (Java, C++, etc.). L'idée que nous émettons est la suivante : plus le langage informatique s'est détaché du langage machine, plus la distinction entre libre et fermé a eu du sens.

2.3.3. Le développement de la micro-informatique.

Le développement de la micro-informatique contribua également à l'apparition de la production distribuée de logiciels. Le rôle joué par le micro-ordinateur fut similaire à celui que le métier à tisser joua dans l'industrie domestique. Il est donc peu concevable d'avoir une production de logiciels distribuée sans l'ordinateur personnel. Le premier micro-ordinateur fut inventé en 1975 par Steve Jobs et Stephen Wozniak, et se nommait l'Apple I. Il est très rapidement suivi par l'Apple II. IBM a lancé son PC en 1981, aujourd'hui devenu un standard de fait.

2.3.4. Un moyen de communication à un coût nul ou quasi-nul.

A. Le rôle d'Arpanet[69].

« Arpanet est un réseau de commutation de paquets qui permet d'assurer l'acheminement des données informatiques sans qu'il existe sur ce réseau un ordinateur central qui en commande tout le fonctionnement. » (FUNOC 2009) La création d'Arpanet marque le passage des systèmes centraux aux systèmes répartis. Arpanet est un réseau développé en 1969 par R. Taylor de l'agence ARPA, celui-ci avait pour objectif d'assurer une meilleure communication entre les chercheurs afin que ces derniers se détachent de la dépendance des grandes entreprises de l'industrie informatique. Au départ, Arpanet reliait quatre ordinateurs situés dans des centres universitaires, puis ce nombre a rapidement grimpé. En 1973, il avait atteint trente cinq.

En 1983, Arpanet se divisa en deux parties : l'une réservée à la défense américaine, l'autre aux chercheurs. L'idée d'Arpanet était de contourner la disparité des systèmes en adoptant un

[69] L'histoire d'internet qui vous est présentée est largement basée sur la très bonne chronologie proposée par le site FUNOC (**F**ormation pour l'**Un**iversité **O**uverte de **C**harleroi).

langage de communication commun à tous les ordinateurs du réseau. Le premier protocole NCP XXX fut mis au point en 1970.

B. Le développement du TCP/IP.

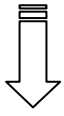
Le second protocole de communication développé afin de contourner l'hétérogénéité des systèmes est le TCP/IP (Transmission Control Protocol/Internet Protocol). Le TCP/IP fut conçu en 1974 par l'Inter Network Working Group, alors dirigé par V. Cerf. Etant donné que pour utiliser Arpanet, les chercheurs devaient recevoir l'accord du Département de la Défense américain, les chercheurs en informatique eurent l'idée de concevoir CNET (Computer Science Research Network) en 1979. D'autres réseaux furent développés par des sociétés commerciales, il y avait notamment BITNET (Because It's Time Network) d'IBM, l'UUCP (UNIX to UNIX Copy Protocol) de l'AT&T.

C. L'avènement d'internet.

Internet est en fait le réseau des réseaux puisque grâce au protocole TCP/IP (libre et gratuit), il met en relation les différents réseaux informatiques initialement créés. Internet est donc à la fois un produit du monde open source et son moteur. D'après Deek et McHugh, l'open source a construit lui-même les briques et outils nécessaires pour son expansion (Deek et McHugh 2007: 11) dont internet fait partie intégrante. Il n'est donc pas étonnant que les logiciels les plus innovants de l'open source soient principalement dans les domaines de la communication et de la programmation.

Question de Recherche 1 :

QR 1 : D'où provient le modèle racine à l'origine du premier logiciel libre ?



Thèse 1 :

TH 1 : **La communauté d'utilisateurs-développeurs s'appuie : d'une part sur un système de management de la solidarité ouvert où les individus sont unis par une solidarité de conception, d'apprentissage et d'usage ; d'autre part sur un système de production distribué qui nécessite un faible investissement capitalistique et dont l'objet réalisé a des propriétés fortement agglutinantes.**

Dans la seconde partie de notre travail, nous avons tout d'abord vu que les travaux sur la caractérisation des communautés se regroupaient en trois cadres d'analyse : les communautés de pratiques et épistémiques, la collective-invention et les réseaux coopératifs. (1) Nous avons montré que la communauté d'utilisateurs-développeurs n'était pas une communauté de pratique mais que celle-ci avait toutefois une dimension épistémique. (2) Puis, nous avons mis évidence l'inadaptation de la notion d'invention pour nommer le phénomène décrit par Allen puisqu'il s'agissait d'un cas d'amélioration d'outil de production conformément à des critères de performance prédéfinis (hauteur et température). En outre, nous avons constaté que la communauté d'utilisateurs-développeurs avait deux différences avec les cas collective-invention : d'une part, dans la collective-invention il y a une uniformisation des objectifs alors que dans les communautés open source les participants ont des objectifs divergents ; d'autre part, Osterloh et Rota (2007) ont souligné que lorsqu'une technologie passait du stade de l'exploration à l'exploitation[70] les groupes de collective-invention disparaissait tandis que les communautés open source existent toujours malgré le fait que l'open source soit passé au stade de l'exploitation. (3) Enfin, la littérature sur les réseaux de collaboration soulignait que la production non-marchande avait pris une ampleur particulièrement importante et plus encore sur internet. Dans ce cadre, la production non-marchande en ligne a reçu plusieurs qualificatifs : elle est tantôt nommée *« peer-production »* (Benkler et Nissenbaum 2006), *« social production »* (Benkler 2006) ou *« réseau coopératif »* (Grassineau 2009). D'autre part, certains auteurs affirment que le modèle des communautés open source n'est pas un nouveau modèle productif du point de vue de l'histoire des systèmes de production (Dalle et al. 2005: 397).

Dans les deux chapitres suivants, nous avons prouvé que la communauté d'utilisateurs-développeurs était issue de la combinaison inédite entre un système de management de la solidarité et un système de production distribué [Thèse 1]. Nous démontrons également que les notions de solidarité et de distribution ne sont pas restreintes à l'open source puisqu'il existe d'autres systèmes de management de la solidarité (SMS) ainsi que d'autres systèmes de production distribués (SPD).

Basé sur ces deux chapitres, nous définissons la communauté d'utilisateurs-développeurs comme un système de management de la solidarité constitué d'acteurs géographiquement distribués à la fois liés par une solidarité d'usage (utilisateurs), d'apprentissage et de conception (développeurs). Ce système produit un objet virtuel et confère des droits d'auteurs

[70] Sur les notions d'exploitation et d'exploration se référer à (Chanal et Mothe 2005).

à ceux ayant concouru à son élaboration et des droits d'utilisation à l'extérieur du collectif par le biais d'une licence.

Nous ajoutons que *le modèle racine* (la communauté d'utilisateurs-développeurs) a connu différentes expansions et hybridations donnant naissance à ce que nous nommons les régimes de l'open source. Par *régimes de l'open source*, nous faisons référence aux éléments qui caractérisent à la fois : le processus de conception, l'exploitation marchande et les propriétés des logiciels libres.

PARTIE III. DU MODELE RACINE A LA VARIETE DES ORGANISATIONS : L'EMERGENCE DES REGIMES DE L'OPEN SOURCE.

L'objet de cette partie est d'offrir une description des différentes expansions du modèle racine à l'origine des régimes de l'open source. Pour ce faire, nous exposerons la variété des organisations de l'open source à la fois des points de vue théorique et pratique [Chapitre 1]. Sur le plan théorique, nous mettrons en évidence le manque d'une typologie des organisations de l'open source. Sur l'aspect pratique, nous rendrons compte de l'hétérogénéité des formes organisationnelles rencontrées sur le terrain. Puis en articulant théorie et pratique, nous modéliserons les régimes de l'open source [Chapitre 2].

Chapitre 1. DE LA VARIETE DES ORGANISATIONS OU L'EMERGENCE DES REGIMES DE L'OPEN SOURCE.

Dans ce premier chapitre, nous détaillerons en premier lieu les fondements théoriques et pratiques des régimes de l'open source [1.]. Nous discuterons pour cela les limites de la notion de *« community »* dans la distinction des organisations de l'open source [1.1.]. Puis, nous exposerons la méthode adoptée pour notre échantillonnage [1.2.]. Enfin, nous détaillerons les critères employés pour la sélection des cas [1.3.].

En second lieu, nous décrirons les cinq cas sélectionnés [2.], à savoir : Kexi, une communauté d'utilisateurs-développeurs assurant le développement d'un module de gestion de bases de données [2.1.] ; Freeworks, une association gérant différentes initiatives autour du concept du libre [2.2.] ; Mandriva, une société française éditrice d'une distribution Linux [2.3.] ; OpenOffice.org, une communauté de firmes assurant le maintient d'une suite bureautique [2.4.] ; OW2 Consortium, un consortium industriel travaillant sur les problématiques du middleware [2.5.]. La présentation de ces cas sera axée sur quatre éléments : le type d'acteurs, le mode de financement, le système de rémunération et la structure juridique.

1. Les fondements théoriques et pratiques des régimes de l'open source.

1.1. Les limites de la notion de « community ».

Dans nos recherches empiriques, nous avons été frappés par la richesse des formes organisationnelles existantes dans l'open source. Or, la littérature utilisait toujours le terme communauté pour désigner ces formes organisationnelles. Par exemple, O'Mahony souligne qu'un *« projet géré par une communauté est un projet open source ou free software initié et géré par un groupe distribué de personnes qui ne partagent pas un même employeur. »* (O'Mahony 2003: 1179). Dans un autre article, O'Mahony soutient que de *« plus précises distinctions sont nécessaires pour communiquer les frontières entre firme, organisations à but non lucratif et communauté »* (O'Mahony 2007: 144).

Ce problème de pluralité fut également mis évidence par Henkel. Henkel propose de distinguer les *« projets »* open source en fonction de leurs participants. Il distingue trois types de projets : (1) ceux uniquement constitués d'acteurs individuels, (2) ceux disposant d'une firme centrale et (3) ceux composés uniquement par des entreprises. Les travaux de Henkel ont donc prouvé qu'il était possible d'avoir une communauté d'entreprises (Henkel 2003a: 3). Plus spécifiquement, Henkel identifia une première dimension permettant de distinguer les organisations de l'open source : le type de participants. Les notions de *projet*[71] et de *communauté* étant bien trop vagues pour distinguer les différentes formes d'organisations dans l'open source.

Depuis ses débuts, l'open source a connu de profondes mutations du point de vue de la nature des organisations comme le souligne un expert de l'open source, *« il y a eu des individuels bénévoles tout seuls dans leur coin. Il y a eu des organisations d'individuels et maintenant ce sont des organisations d'organisations. C'est l'open source de troisième génération*[72]*. »*

[71] Il convient de noter que la notion de *« projet »* dans la littérature sur l'open source n'a pas grand-chose à voir avec la littérature sur la gestion de projet (Garel, Giard, et Midler 2001; Garel 2003b, 2003a; Giard 1991; Giard et Midler 1993; Jolivet 1998; Royer 2005). Par exemple, le triptyque coût, qualité et délai ne se retrouve pas dans les « projets open source ».

[72] Julie Marguerite, Open Source Architect, Thales D3S, entretien avec l'auteur, 6 juin 2007.

1.2. La technique de l'échantillonnage théorique.

La technique de l'étude de cas consiste à choisir délibérément des cas suffisamment hétérogènes pour rendre compte de la variété d'un phénomène (Eisenhardt 1989). Sur la base de cette technique, nous avons sélectionné nos cas grâce auxquels nous avons bâti notre typologie des régimes de l'open source. Nous nous focaliserons particulièrement sur les formes existantes entre d'un côté la communauté d'utilisateurs-développeurs (non rémunération des producteurs et objet produit libre d'utilisation) et la firme (rémunération des producteurs et objet produit valorisé directement ou indirectement).

D'après Christensen, les théories basées sur un grand échantillon n'ont pas une validité externe plus grande que les théories basées sur l'étude d'un nombre limité de cas (Christensen 2006: 53). C'est pourquoi les analyses découlant de nos cas seront suffisamment générales pour expliquer une large partie du phénomène étudié.

Nous ajoutons que le rôle des firmes dans l'open source ne se limite plus au simple financement de communautés ou à la libération de logiciels, les firmes financent elles-mêmes les communautés qu'elles ont créées entre-elles comme les travaux de Boiteux (2002) et Henkel (2003b, 2003a, 2004, 2006) le montrent ou encore Demazière et al. en faisant référence à des organisations constituées *« d'institutions diverses (entreprises, centres de recherche,…) »* (Demazière, Horn, et Zune Forthcoming: 7) sans toutefois proposer une analyse de ces organisations. Les cas OW2 consortium et OpenOffice.org (décrits de manière détaillée ci-après) sont particulièrement significatifs de ce phénomène. Cette variété d'organisations est donc à la fois rencontrée sur le terrain et mentionnée par différents auteurs. En revanche, il n'existe pas de modèle général permettant de distinguer les organisations de l'open source à partir de critères empiriques et encore moins de suivre les transformations organisationnelles d'une forme à une autre.

Il faut tout de même garder à l'esprit que lorsque la notion de communautés d'utilisateurs-développeurs est citée, on pense le plus souvent à des communautés constituées de plusieurs centaines de personnes. Or, la grande majorité des communautés sont d'une taille réduite d'après les résultats de l'étude de Krishnamurthy (2002: 4) dans laquelle il s'est intéressé à 100 logiciels libres du site SourceForge.net. Il trouve une moyenne de 6,61 développeurs par logiciel, une médiane à 4 et un mode[73] de 1.

[73] Dans une série statistique, le mode est la valeur retrouvée le plus fréquemment.

D'après Rannou et Ronai, *« la majorité des projets est élaborée par 1 développeur (28,4% des projets), 2 développeurs (24,3% des projets) ou 3 développeurs (24,3% des projets) »* (Rannou et Ronai 2003a: 127), ces données furent obtenues à partir de l'étude de 16 905 logiciels libres. Par conséquent, il est difficile d'accepter que les logiciels libres soient développés par de grandes communautés comme le prétendent certains auteurs (Bonaccorsi et al. 2006: 1086).

Le but de notre travail était de bâtir une typologie des formes organisationnelles existantes dans l'open source. D'après Eisenhardt et Graebner, le choix d'une approche qualitative basée sur de multiples cas vise à la construction de théories. De ce fait, nous ne pouvions pas avoir une approche quantitative puisque ces approches visent à tester des hypothèses. Nous avons choisi d'étudier les organisations de l'open source en nous basant sur de multiples études de cas en utilisant la technique de l'échantillonnage théorique ou *« theoretical sampling »*. Dans cette méthode, la sélection des cas peut être basée notamment sur la réplication, l'extension ou l'inversement d'un phénomène (Eisenhardt et Graebner 2007: 27).

1.3. Les critères de sélection des cas.

Pour réaliser notre sélection, nous avons défini des critères à la fois appuyés sur la variété des formes rencontrées sur le terrain, les notions de solidarité et de distribution [Partie II] et la littérature existante sur les organisations de l'open source.

Le premier critère est le type d'acteurs puisque la littérature fait état de plusieurs catégories d'acteurs dans les organisations de l'open source (Henkel 2003a: 3; O'Mahony 2003: 1180; 2007: 144). Nous avons résumé les types d'acteurs, les liens de solidarité et les cas étudiés correspondants dans le tableau ci-dessous.

Tableau 13 : Types d'acteurs dans les organisations de l'open source.

Types d'acteurs[74]	Sources	Liens de solidarité	Cas (mineurs et majeurs)
Utilisateurs (U)	(Demazière et al. 2007: 40; Hertel et al. 2003: 1165)	Usage (utilisateurs) [Partie II].	Claroline/Dokeos, Kexi, Freeworks.
Développeurs (D)	(Demazière et al. 2007: 40; Hertel et al. 2003: 1165)	Conception et apprentissage (développeurs) [Partie II].	Mandriva, Mozilla, OpenOffice.org, OW2 Consortium, Apache.
Utilisateurs-Développeurs (UD)	(Baldwin et Clark 2005: 33; Gaudeul 2007: 242; Hicks et Pachamanova 2007: 316; Lerner et Tirole 2000: 18; West et O'Mahony 2005: 1)	Usage, conception et apprentissage [Partie II].	Kexi, Freeworks, Mandriva, Mozilla, Adonthell, TDR, Claroline/Dokeos, NeoOffice, Apache.
Entreprises (E)	(Boiteux 2002: partie 5; Henkel 2003b, 2003a)	Economique (mutualisation de coûts et/ou lutte contre monopole), technologique (maintien d'une base de code générique commune : composant ou commodité).	Thales, QualiPSo, OW2 Consortium, OpenOffice.org.

Le second critère correspond au mode de financement de l'organisation. Les différents modes de financements peuvent être résumés dans le tableau ci-dessous.

Tableau 14 : Modes de financement dans les organisations de l'open source.

Types de financement	Principe	Sources	Cas (mineurs et majeurs)
Propres Ressources (PR)	Utilisation de ressources privées pour construire un bien collectif.	(von Hippel et von Krogh 2003: 213)	Kexi, Freeworks, Mandriva, Mozilla, Adonthell, TDR, Claroline/Dokeos, NeoOffice, Apache.
Partage de Coûts (PC)	Mutualisation des coûts de développement sur composants génériques communs ou commodités.	(Boiteux, 2002: partie 5;(Deek et McHugh 2007: 9)	OpenOffice.org, OW2 Consortium.
Sponsors (S)	Recours à des sponsors pour le financement de matériel ou d'événements.	(Dahlander et McKelvey 2005: 634; O'Mahony 2003: 1180),	Kexi, OpenOffice.org, Apache.
Prestation de Services[75] (PS)	Adaptation du modèle économique aux logiciels libres.	(Bonaccorsi et Rossi 2003a: 1249)	Thales, Mekensleep, Wallix, Mozilla.
Dons (D)	Réalisation de campagnes de dons auprès des membres et des utilisateurs.	(Fitzgerald 2007: 591; Klincewicz 2005: 15).	Kexi, Apache, Mozilla, Freeworks.

[74] La littérature fait aussi état de contributeurs sponsorisés (O'Mahony, 2003 ; West et O'Mahony, 2005 ; 2007) mais il ne s'agit pas d'une catégorie d'acteur particulier puisqu'il ne s'agit de représentants d'une entreprise dans une communauté existante. L'entreprise n'existe que par ses représentants puisque par définition ce n'est qu'une fiction juridique.

[75] Nous avons volontairement laissé une appellation générale pour incorporer la plus grande partie des modèles économiques liés aux logiciels libres.

Le troisième critère concerne le système de rémunération puisqu'il existe plusieurs cas de figure résumés dans le tableau ci-dessous.

Tableau 15 : Modes de rémunération dans les organisations de l'open source.

Mode de rémunération	Principe	Sources	Cas (mineurs et majeurs)
Bénévolat (B)	Absence de rémunération que se soit de manière directe ou indirecte.	(Bitzer et Schröder 2005: 390; Kogut et Metiu 2001a: 249)	Kexi, Freeworks, Mandriva, Mozilla, Adonthell, TDR, Claroline/Dokeos, NeoOffice, Apache.
Rémunération Directe (RD)	Les développeurs sont rémunérés pour leur travail.	(Crémer et Gaudeul 2004: 115; Dahlander et Magnusson 2005: 483; Ghosh et al. 2002: 63)	Kexi, Mozilla, OpenOffice.org, OW2 Consortium, Apache, Thales, Wallix, Mekensleep.
Rémunération Indirecte (RI)	Les développeurs perçoivent une rémunération indirecte par le biais de services liés au logiciel libre.	(Benkler 2002: 438)	Freeworks, Mandriva, OpenOffice.org, OW2 Consortium.

Le quatrième critère porte sur l'existence d'une structure juridique ou pas : la littérature fait état de structures *Informelles* (I) (communautés) et *Formelles* (F) (les firmes (Bonaccorsi et Rossi 2003b: 12) et les fondations (O'Mahony 2002: 21)).

Il est largement reconnu qu'il existe une grande variété de formes organisationnelles développant des logiciels open source (Shah 2006: 1001). Les notions de *« Bazaar »* et de *« community »* renseignent peu sur la manière dont ces collectifs fonctionnent réellement (Demazière et al. 2007: 35). La question qu'il faut se poser est qu'est-ce qu'une communauté ? Est-ce qu'un groupe de bénévoles développant un jeu vidéo sur leur temps libre et un consortium rassemblant plus de 60 entreprises et instituts de recherche autour des problématiques du Middleware peuvent-ils être tous deux nommés communautés ? Bien au contraire, nous pensons qu'il est nécessaire de réaliser une typologie permettant de faire la distinction entre les différentes organisations de l'open source.

2. L'hétérogénéité des organisations de l'open source : une étude empirique.

Nous avons choisi de mener une approche comparative basée sur de multiples cas. Ces cas furent choisis en maximisant leurs différences à partir de quatre critères : (1) le type d'acteurs, (2) le mode de financement de l'organisation, (3) le système de rémunération des participants

et (4) la structure juridique. Les cas sélectionnés sont les suivants : (1) Kexi, une communauté d'utilisateurs-développeurs assurant le développement d'un système de gestion de base de données ; (2) Freeworks, une association de loi 1901 gérant différentes initiatives autour du concept du libre et en particulier un système d'administration de base de données ; (3) Mandriva, une société française éditant une distribution Linux et gérant une communauté d'utilisateurs-développeurs contribuant à ce logiciel ; (4) OpenOffice.org, une communauté rassemblant principalement des firmes de l'industrie informatique assurant le maintien d'une suite bureautique à environnements multiples ; (5) OW2 Consortium, un consortium industriel rassemblant plus de 60 entreprises et instituts de recherche travaillant sur les problématiques du middleware libre. Nous réaliserons une monographie de chaque cas en nous focalisant sur les quatre critères précédemment précisés.

2.1. Le cas Kexi ou un exemple du modèle racine.

2.1.1. L'histoire de Kexi.

Kexi est un logiciel de gestion de base de données avec une interface graphique. Le développement de Kexi débuta en fin d'année 2002 par un jeune développeur autrichien que nous nommerons Fondateur 1[76]. Kexi fut conçu car : *« Le segment occupé par Kexi est quasiment vide en ce qui concerne le libre, et l'idée de développer une alternative libre à MS Access est assez répandue, étant donné que ce logiciel représente une des raisons pour de nombreuses entreprises de rester sous Windows. »* (Développeur Actif 1, Kexi).

Ce logiciel fut lancé pour combler un manque portant sous l'environnement Linux. Initialement, le logiciel n'était pas conçu avec une structure en modules et cela était particulièrement problématique pour les autres développeurs. En 2003, un nouveau développeur (Fondateur 2) proposa de réécrire Kexi en adoptant une structure modulaire. Depuis 2003, Fondateur 1 ne participe plus au développement de Kexi.

2.1.2. Les développeurs de Kexi.

Kexi est composée de trois types d'acteurs. (1) Les développeurs fondateurs : ils ont marqué le développement de Kexi par leurs importantes contributions mais ils ne sont plus

[76] Pour des raisons de préservation de la confidentialité, nous utiliserons des appellations génériques pour chaque participant dans l'ensemble de nos cas.

actifs en développement. (2) Les développeurs actifs : ils maintiennent actuellement Kexi. (3) Les développeurs moins actifs : ils participent ponctuellement au développement de Kexi ; en revanche, leurs contributions sont irrégulières. Nous décrirons les développeurs clés de Kexi afin d'illustrer la composition de cette organisation en mettant en avant leur rôle respectif et les raisons de leur participation.

A. Fondateur 2.

Fondateur 2 est le seul développeur à percevoir une rémunération pour le travail qu'il réalise sur Kexi. Il est employé par une société proposant des services associés à Kexi et des versions pré-packagées comprenant des *addons*[77] non libres car la licence de Kexi (LGPL) le permet. Après le départ de Fondateur 1 en 2004, Fondateur 2 est devenu le leader de cette communauté. Fondateur 2 base sa légitimité sur deux éléments : la quantité et la qualité de ses contributions au code source. Le tableau ci-dessous présente la contribution de chaque développeur au code source de Kexi.

Tableau 16 : Répartition des contributions à Kexi.

Contributeurs	Fichier copyrightés	Contribution[78]
Fondateur 2	481	71%
Fondateur 1	131	19%
Développeur actif 1	121	18%
Développeur actif 2	103	15%
Ex-Développeur Actif 1	42	6%
Ex-Développeur Actif 2	16	2%
Ex-Développeur Actif 3	11	2%
Ex-Développeur Actif 4	9	1%
Ex-Développeur Actif 5	7	1%
Total des fichiers copyrightés	674	-

Nous pouvons noter que Fondateur 2 a le pourcentage de fichiers copyrightés le plus important. D'autre part, pour bien souligner le fait que Fondateur 2 est considéré comme le leader de Kexi, nous vous citerons les propos de quelques développeurs.

« De mon point de vue, je pense que Fondateur 2 est une sorte de leader, car il travaille à plein temps sur Kexi, et car il a des intérêts commerciaux [vis-à-vis du logiciel] dans la société qu'il l'emploie… » (Développeur Actif 1, Développeur actif, Kexi).

[77] Un add-on est un module complémentaire.
[78] On notera le fait que la somme des contributions est supérieure à 100%, cela s'explique tout simplement par le fait qu'un même fichier peut avoir plusieurs copyrights.

« Il est clair (au moins pour moi) que Fondateur 2 aura toujours le dernier mot… » (Développeur Actif 2, Développeur actif, Kexi).

« Fondateur 2 est le plus âgé, le plus expérimenté, etc. […] en logiciels libres, la seule légitimité c'est le code que vous écrivez. » (Développeur Actif 1, Développeur actif, Kexi).

B. Développeur Actif 1.

Développeur Actif 1 est étudiant à l'Ecole Polytechnique. Il se définit lui-même comme : *« Un développeur actif pour Kexi depuis décembre 2003. Je m'occupe de l'écriture du module de création de formulaires et de l'éditeur de propriétés. J'ai aussi participé à la traduction en français du logiciel (avec l'équipe française de traduction de KDE). »* (Développeur Actif 1, Développeur actif, Kexi).

Développeur Actif 1 relate l'histoire de sa participation à Kexi : *« En septembre 2003, j'ai commencé à contribuer au projet KDE par la traduction de certains logiciels, tout en commençant l'apprentissage de la programmation KDE. J'ai ensuite écrit un petit patch pour Kaboodle (le lecteur multimédia de KDE). Une fois ma décision de contribuer prise, j'ai cherché un projet qui m'intéressait et je suis tombé sur Kexi. J'ai été très choqué de voir qu'un projet d'une telle importance pour l'adoption de Linux avait si peu de développeurs. J'ai donc contacté [Fondateur 1], puis nous avons discuté sur IRC pour savoir ce que j'allais faire. Comme je ne connaissais pas grand chose aux bases de données, il ne restait plus que les formulaires. J'ai trouvé l'idée plutôt intéressante et je me suis lancé là dedans, malgré ma quasi totale inexpérience en développement. »* (Développeur Actif 1, Développeur actif, Kexi).

Développeur Actif 1 justifie sa participation en ces termes : *« Je pense que Kexi peut avoir une très grande importance dans l'adoption de systèmes totalement libres comme Linux par les particuliers et les petites entreprises, qui utilisent souvent des logiciels comme MS Access. Kexi est là pour combler une lacune pour moi très importante, et c'est pourquoi j'ai choisi ce projet. »* (Développeur Actif 1, Développeur actif, Kexi).

Développeur Actif 1 parle de la manière suivante des autres membres de Kexi : *« Ils sont tous plus âgés et ont plus d'expérience que moi donc je les respecte énormément. [Fondateur 2] et [Développeur Actif 2] sont aussi devenus des amis avec le temps. »* En outre, il

souligne : *« Il y a une grande entraide au sein de notre projet, et c'est ce qu'il fait qu'il progresse bien. »* (Développeur Actif 1, Développeur actif, Kexi).

C. Développeur Actif 2.

Développeur 2 se décrit lui-même de la manière suivante : *« Je suis un gars âgé de 28 ans habitant Berlin en Allemagne. […] J'étudie après le travail ces dernières années en tant que développeur logiciel. Bon, en dehors d'apprendre j'aime juste apprécier la vie. J'aime ma famille, mes amis et tous ceux qui me procurent un bon carma. Apprendre est une des plus belles choses que la vie propose. Cela conserve l'esprit propre et fait plaisir. Donc apprendre était et est toujours une de mes premières priorités dans la vie. J'aime planifier et organiser des choses. C'est bien d'avoir des choses qui marchent d'une manière efficiente et de suivre une cible définie par soi-même. Communiquer avec les autres, échanger des opinions et apprendre comment les autres font les choses sont important pour moi. »*

« Je suis le développeur et le responsable de Kross, situé dans koffice/libs/kross, il s'agit de notre framework de scripts à l'intérieur de KOffice. En dehors de ça, j'ai écrit le python-binding, résolu un grand nombre de bugs en général dans KOffice et je travaille environ depuis deux ans sur Kexi aussi. En ce moment, mon travail est KoMacro, un macro-framework pour Kexi. » (Développeur Actif 2, Développeur actif, Kexi).

Développeur Actif 2 explique l'histoire de sa participation à la communauté : *« Quelques années auparavant, j'ai commencé à développer KMLDonkey qui plus tard a été intégré dans kdeextragear. Un jour j'ai réalisé que KMLDonkey était complet au niveau fonctionnalités et il avait tous ce que je pouvais espérer qu'un logiciel de ce type fasse. Déjà quelques années auparavant je recherchais des alternatives libres à MSAccess et j'ai échoué à trouver une bonne solution. Donc, le meilleur en terme de potentiel et meilleur en terme de code source parmi les projets libres autour était Kexi. Et donc, j'ai finalement décidé de les aider à améliorer le logiciel. »* (Développeur Actif 2, Développeur actif, Kexi).

Développeur Actif 2 souligne qu'il participe à Kexi : *« Parce que c'est une manière utile et bonne d'améliorer le projet d'un côté et d'améliorer mes compétences de l'autre. »* (Développeur Actif 2, Développeur actif, Kexi).

Les développeurs de Kexi sont liés par une solidarité d'usage car Fondateur 2 a dit : *« Nous sommes tous enrôlés dans ce projet en tant qu'utilisateurs [de Kexi] (ou futur*

utilisateurs)... » D'autre part, il a ajouté : *« Nous avions tous besoin de quelque chose comme cela [Kexi][79], nous n'étions pas satisfaits par les projets existants, et nous nous sommes rencontrés... Donc nous avions un objectif commun. »* (Développeur Actif 2, Développeur actif, Kexi).

Les développeurs de Kexi sont aussi liés par une solidarité d'apprentissage. Ces derniers s'expriment en ces termes : *« Un autre point positif pour les discussions par le biais de l'IRC est le fait que la majorité des problèmes que nous rencontrons lorsque nous développons peuvent être résolus ensembles, et de ce fait nous apprenons des uns et des autres. »* (Développeur Actif 1, Développeur actif, Kexi).

2.1.3. Le système de rémunération et financement de Kexi.

L'ensemble des développeurs de Kexi ne sont pas rémunérés pour le travail qu'ils réalisent sur Kexi excepté Fondateur 2 employé à plein temps pour travailler sur Kexi. A la question : *« existe-il un système de rémunération ? »* posée sur l'IRC de Kexi les membres de Kexi ont répondu :

<Développeur Actif 1> Nordine[80] : *« Pas vraiment »*

< Développeur Actif 1> Nordine : *« un magasine français a proposé de nous donner de l'argent, mais du fait qu'il n'y a pas d'organisation [du point de vue] légal cela peut être un problème »*

< Développeur Actif 1> Nordine : *« et c'est dur de choisir c'est qui peut prendre l'argent, etc. »*

<Fondateur 2> Nordine : *« Excepté les fois où Google contribuera avec de l'argent »*

< Développeur Actif 2> : *« Excepté de voir et d'aider à produire un fantastique projet, améliorer nos propres compétences, travailler sur quelque chose de vraiment utile, … »*

(Développeur Actif 1, Développeur Actif 2, Fondateur 2, Développeurs actifs, Kexi).

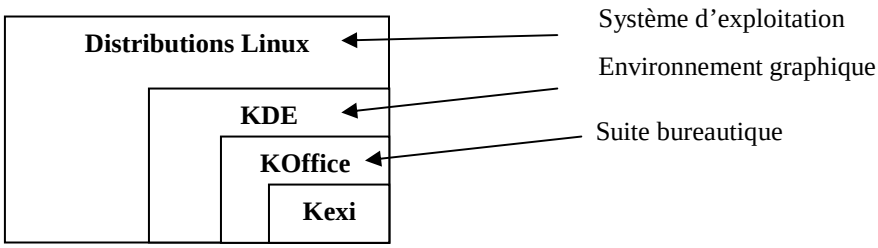
Kexi est financé par des dons réalisés par des partenaires et des utilisateurs par le biais de l'association KDE e.V.. D'après Fondateur 2 : *« [...] L'infrastructure comme les serveurs KDE sont fournis par KDE e.V. avec le soutien par exemple d'universités, de distributeurs, etc. »* (Fondateur 2, Développeur actif, Kexi).

[79] Les éléments entre crochets sont nos ajouts visant à améliorer la compréhension du lecteur.
[80] Sur l'IRC, il convient de préciser le nom de la personne à qui l'on souhaite s'adresser étant donné le nombre de personnes sur le Chat.

« Kexi n'est pas une organisation formelle mais celle-ci s'insère dans KOffice, qui elle aussi n'est pas une organisation formelle. [En revanche], KDE e.V. [une forme d'association à but non lucratif en Allemagne] est une organisation très formalisée et très connue. » (Fondateur 2, Développeur actif, Kexi).

De manière schématique l'environnement de Kexi peut-être simplifié dans le schéma ci-dessous.

Figure 15 : Schéma de l'environnement technique de Kexi.



2.2. Le cas Freeworks : un modèle associatif.

2.2.1. L'histoire de Freeworks.

FlashMyAdmin est le *« projet déclencheur »* (expression de Fondateur 1) de cette organisation. C'est une application d'administration MySQL en Action Script (AS). FlashMyAdmin fut inspiré d'un logiciel existant : PhpMyAdmin. Le fondateur recherchait à résoudre un problème ergonomique lié à la gestion de base de données. Selon lui : *« Le problème avec PhpMyAdmin est que l'utilisateur ressent les processus d'échanges entre client et serveur. L'idée était donc d'exploiter l'avantage d'avoir un logiciel en ligne et l'ergonomie d'un logiciel installé sur le poste de travail. »* (Fondateur 1, Freeworks).

La première version de FlashMyAdmin a été développée par un seul développeur en 2002, il s'agit de Fondateur 1. Il relate l'histoire de Freeworks : *« Ca débute au départ […] en 2003. Même, peut-être même fin 2002, je crois… Je réserve le nom de domaine FlashMyAdmin. FWO [Freeworks] n'existe pas. FlashMyAdmin n'existe pas la seule idée c'est de faire un logiciel et dans la minute qui suit après de le faire libre. Donc il y a juste l'idée de faire un logiciel libre je commence à le faire et tout, je suis tout seul etc. Après en 2004, à cette époque là je travaille. Je travaille en tant qu'employé quoi donc je le fait le soir et le week-end. Ensuite en 2004, avec un ami musicien et informaticien nous faisons Freeworks. Et*

ensuite le projet [FlashMyAdmin] c'est essentiellement moi mais il y a Olivier qui participe à la partie graphique, et après plus récemment, il y a Paul qui s'occupe de la réécriture. » (Fondateur 1, Freeworks).

A la question : Pourquoi le fondateur du logiciel a-t-il fait le choix d'offrir le code source au lieu de le garder secret ? Le fondateur propose : *« Une réponse pragmatique en inversant le problème : la production d'un logiciel propriétaire était au dessus de [m]es moyens ! (brevets, monter une société et embaucher pour assurer un développement rapide, sauvegarde de l'idée, etc.). Un tel choix implique d'être fortement motivé par un retour financier.*

Si vous êtes seulement motivé par : Utiliser ce logiciel pour vous-même et avoir le plaisir de l'écrire ; Penser qu'il peut-être utile à d'autres ; Souhaiter quand même le retour & avis des utilisateurs et le réaliser comme un soft professionnel quant au résultat (pas les moyens) ; Penser et apprécier qu'il puisse dans le futur avoir une vie propre qui vous dépassera (ce qui vous rendra disponible pour un autre projet …) […]. Alors il faut mieux sans contrainte et spontanément démarrer un projet libre. » (Fondateur 1, Freeworks).

Quelques mois après l'écriture des premières lignes de code, l'association Freeworks fut créée afin de remplir quatre objectifs : développer, faire connaître, organiser la contribution et financer le développement de FlashMyAdmin. Freeworks est une association régie par la loi du 1er juillet 1901. Fondateur 1 décrit Freeworks comme : *« Une association plutôt intimiste d'auteurs/contributeurs (9 membres) souhaitant se consacrer à la réalisation de travaux « libres » et les faire connaître. »* (Fondateur 1, Freeworks).

Freeworks assure le développement d'autres *« travaux libres*[81] *»* comme XmediaPlayer, un lecteur multimédia capable de lire des images, du html, des animations, des mp3 et des vidéos. XmediaPlayer est à l'origine de deux initiatives : Freewave, une radio diffusant de la musique libre et Freegallery, un site d'art plastique libre. Freeworks diffuse également FunSquare, un jeu de casse-tête dans le même style que le Rubik's cube en deux dimensions. Une version de FunSquare est prévue pour les téléphones portables. C'est un jeu gratuit mais non libre, selon Fondateur 1, le code source de ce jeu n'a pas tellement d'intérêts pour les utilisateurs.

Au moment de notre première interview, Fondateur 1 et Développeur-Musicien 1 était sur le point de réécrire entièrement le code de FlashMyAdmin en Action Script 2. Les raisons mentionnées étaient d'une part pour réorganiser le code source en modules et d'autre part pour pouvoir faire évoluer FlashMyAdmin plus facilement. La réorganisation du code source

[81] Expression de Fondateur 1, Membre très actif, Freeworks.

en modules avait également pour but d'inciter de nouveaux développeurs à contribuer à FlashMyAdmin sans connaître l'ensemble du code source.

2.2.2. Les membres de l'association Freeworks.

Au moment de l'étude Freeworks comprenait 9 membres[82], ils se divisent en trois catégories les membres très actifs, moyennement actifs et peu actifs.

A. Les membres très actifs.

→ Développeur-Musicien 1.

Développeur-Musicien 1 se décrit de la manière suivante : *« Donc, moi si tu veux je assez atypique, car je n'ai pas suivi le cycle classique de l'informatique. A la base, j'étais musicien. Après une tournée un peu ratée… Tu vois c'est un peu dur de gagner de l'argent avec la musique…Je suis devenu informaticien, pur et dur tu vois. Je travaille aussi dans le pas libre quoi, pour bouffer. Je suis une entreprise à moi tout seul, je suis indépendant […] Et je loue mes compétences informatiques à des sociétés. Grâce à cela, j'ai une grosse expérience dans plein de langages différents. J'ai re-rencontré [Fondateur 1] il n'y a pas très longtemps, et j'ai bien aimé son projet : FlashMyAdmin. Je lui proposé de développer un peu FlashMyAdmin. En fait, j'ai participé à une migration de FlashMyAdmin… »* (Développeur-Musicien 1, Membre très actif, Freeworks).

Développeur-Musicien 1 raconte l'histoire de sa participation à Freeworks : *« Jean-Michel était un copain d'école de mon frère. Je l'ai rencontré chez mon frère, un truc comme ça. Et j'ai vu que Jean-Michel était un informaticien à sa manière de parler, de convaincre, d'aller jusqu'au bout des trucs pour que ça marche, on est un peu maniaque en informatique. J'ai vu son projet, et j'ai trouvé ça bien. Moi, j'avais des bases en MySQL donc je connaissais la gestion de bases de données ce que fait le FlashMyAdmin. Il m'a parlé de sa conception du libre, et ça m'a plu. Pour faire du libre, il y a vachement de contraintes à avoir, faut que le code soit bien organisé, qu'il soit expliqué au mieux, etc. C'est plus que professionnel, sinon ça ne marche pas. En gros, il faut habiller le libre. Ensuite, Jean-Michel m'a mis les sources*

[82] Etant donné l'importance du rôle de Fondateur 1 et Développeur-Musicien 1 dans le développement de Freeworks, ces deux acteurs feront l'objet d'une présentation beaucoup plus approfondie. D'autre part, nous n'avons pas eu l'occasion de nous entretenir avec tous les membres de l'association.

en téléchargement et je les ai prises. Puis, on s'est vu deux ou trois fois pour m'expliquer ce qu'il voulait, et j'y suis allé. » (Développeur-Musicien 1, Membre très actif, Freeworks).

Développeur-Musicien 1 décrit son statut dans Freeworks : *« Je suis un membre actif dans l'organisation. Comme, je te l'ai dit j'ai aidé au développement de FlashMyAdmin, cela signifie que pendant quinze jours je n'ai rien fait d'autre que développer. Et cela a redonné un coup de peps à Jean-Michel du coup. Donc on s'auto-dynamise en fait. »* (Développeur-Musicien 1, Membre très actif, Freeworks).

Développeur-Musicien 1 justifie sa participation en ces termes : *« Par plaisir, par amitié pour Jean-Michel, pour apprendre. Jean-Michel m'a appris beaucoup de trucs, et moi en retour je l'ai aidé à upgrader FlashMyAdmin. » […] « Ca me fait plaisir… Je suis assez passionné d'informatique. Je fais aussi de la musique, et pour faire de la musique, il faut être passionné… »* […] *« La manière selon laquelle il était écrit [FlashMyAdmin] ne suivait plus le langage, il a fallu le réécrire entièrement. J'ai remis à niveau en nouvelle version en fait. Je l'ai fait gratuitement, pour le fun, car le produit était sympa. Donc, par ce biais cela m'a permis d'apprendre Flash car Jean-Michel le connaissait beaucoup mieux que moi. »* (Développeur-Musicien 1, Membre très actif, Freeworks).

→ **Graphiste 1.**

Graphiste 1 travaille sur les éléments graphiques de FlashMyAdmin et FunSquare. Il assure la promotion de Freeworks et propose divers conseils. Il participe au débat concernant l'évolution de FlashMyAdmin sans doute car Graphiste 1 réalise la veille technologique qui selon Fondateur 1 : *« Est un don naturel chez lui !!! Il nous envoie des emails, au moins un ou deux par semaine, avec les informations intéressantes qu'il a trouvées. Le travail [de Graphiste 1] est complémentaire du mien […], il trouve une idée qui n'a rien à voir avec notre projet et moi […] je creuse au niveau technique. »* (Fondateur 1, Membre très actif, Freeworks).

→ **Peintre 1.**

Peintre 1 est artiste peintre et poète. Il participe à la mise en place d'une galerie en ligne Freegallery où il expose ses peintures et travaille également sur l'enregistrement de ses poèmes et interviews pour habiller ses reproductions de peintures en ligne.

B. Les membres moyennement et peu actifs.

→ **Développeur 2.**

Développeur 2 est spécialisé dans les techniques pointues de référencement de sites internet. Il s'occupe du référencement des sites de l'association et de FlashMyAdmin.

→ **Auteur-compositeur 1.**

Auteur-compositeur 1 travaille sur Freewave. Il étudie l'informatique avec Fondateur 1 pour pouvoir créer ses morceaux sur ordinateur.

→ **Graphiste 2.**

Graphiste 2 a réalisé l'image centrale du jeu FunSquare. Il travaille également sur l'implémentation du jeu sur les téléphones mobiles puisque le jeu sera multi-images.

→ **Apiculteur 1 :**

Apiculteur 1 est un supporter et c'est aussi l'ami de Fondateur 1. Il travaille sur des textes de formation sur l'apiculture et l'agriculture biologique.

→ **Développeur-Technicien 1 :**

Développeur-Technicien 1 intervient lorsqu'il y a des problèmes de réseaux et d'administration de serveurs Web (Apache, bases de données). Il s'agit d'un ancien collègue de travail de Fondateur 1.

2.2.3. Le système de rémunération et de financement de Freeworks.

Freeworks est une association régie par la loi de 1901. Cela signifie qu'elle n'a pas le droit de réaliser des bénéfices. Il n'existe pas de système de rémunération directe pour la réalisation des ouvrages libres. En revanche, il existe une forme de rémunération indirecte pour les acteurs le souhaitant. Un système de rémunération essayant de faire converger passion et savoir-faire. Le fonctionnement de ce système peut être divisé en quatre parties : (1) les membres de Freeworks recherchent eux-mêmes leurs clients et réalisent les services demandés, (2) Freeworks facture au client la prestation réalisée, (3) le client paye Freeworks, (4) Freeworks verse un salaire au membre pour son travail en prélevant : d'une part 3%

minimum sur la prestation pour son autofinancement, d'autre part Freeworks paye toutes les charges, impôts et prestations sociales prévues par la loi. Néanmoins, étant donné que Freeworks est une association celle-ci n'avance pas d'argent : elle verse uniquement la paye au membre lorsque le client a réglé la facture.

L'association joue donc le rôle de support juridique pour l'activité de ses membres. Son système de fonctionnement, au moins sur ce point, est comparable à celui d'une société de portage salarial. Finalement, le financement de Freeworks provient de deux sources : premièrement des dons et cotisations des membres et deuxièmement des commissions prélevées sur les travaux réalisés (cf. mécanisme décrit ci-dessus).

2.3. Le cas Mandriva : une firme de l'open source.

2.3.1. L'histoire de Mandriva.

Mandriva est une société française commercialisant une distribution Linux. Ce produit s'appelle Mandriva Linux, un système d'exploitation avec une interface graphique nativement intégrée. Cette distribution privilégie la facilité d'utilisation et l'ergonomie. Mandriva Linux est un concurrent direct des autres distributions de Linux, Unix et bien sûr Windows. Le nombre d'utilisateurs de Mandriva Linux est estimé entre 6 et 8 millions. C'est une des versions de Linux qui est considérée comme la plus simple d'utilisation.

D'après l'ex-Directeur Général de Mandriva : *« Mandriva a été fondé en 98 par trois jeunes internautes qui ont créé la société sans même s'être rencontré : Fondateur 1, Fondateur 2 et Fondateur 3. Fondateur 1 avait fait une distribution, je ne sais pas si c'est lui qui a choisi le nom, et il a « rencontré » les deux autres par hasard sur le net, et ils ont décidé de créer une boîte pour la diffuser. »* (Ex-Directeur Général, Mandriva).

Le fondateur initial explique comment il a été amené à créer *« Mandrake »* (ancien nom de Mandriva Linux) : *« Il y a quelques années, en 1998, Linux n'était pas simple du tout à utiliser à moins d'avoir une réelle expérience des systèmes UNIX. Installer un nouveau périphérique signifiait la plupart du temps une recompilation du noyau Linux avec les options adéquates. Utiliser un cd-rom ou un lecteur de disquettes n'était pas possible sans taper des commandes du type "mount -t iso9660 /dev/hdc /mnt/cdrom". Ce qui n'est pas une action triviale pour le néophyte. Les environnements graphiques sous Linux de l'époque étaient à mon sens plutôt des systèmes de "décoration" de l'écran, pour exagérer un peu. C'était sans*

doute une bonne manière de justifier l'utilisation de X-Window, plutôt que toute autre chose :) J'ai donc travaillé plusieurs mois pour créer Linux-Mandrake, seul et contre tous ;) A la maison[83]*. »* (Fondateur 1, Mandriva).

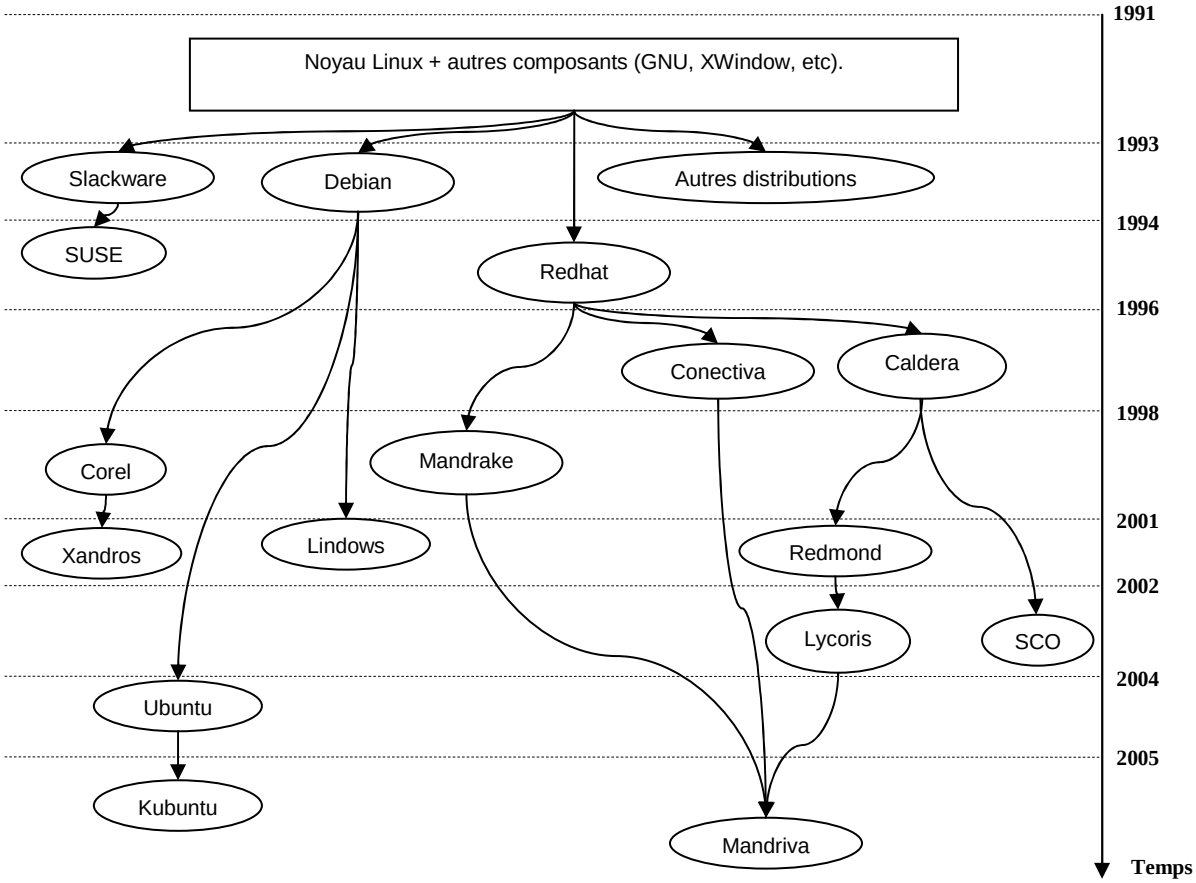
Mandrake Linux était un fork[84] de la version 5.1. de la distribution Red Hat. La plus importante innovation apportée par Mandrake Linux est : *« L'idée de base… qui était de prendre une [distribution] Redhat et de coller KDE [environnement graphique] dessus… C'est une innovation première, etc. c'est une bonne. »* (Ex-Directeur Général, Mandriva).

A partir de 2004, MandrakeSoft était impliqué dans un processus de fusions et d'acquisitions. Ainsi d'après l'ancien Directeur général : *« On a racheté des entreprises, on a racheté en 2004 Edge IT qui était une petite structure française de 5, 6 personnes qui faisait du support Linux, donc ça a été notre premier rachat. Ensuite on a racheté Conectiva en 2005 qui était un plus gros morceau, ça faisait la même taille que nous en nombre de personnes mais pas en chiffre d'affaires, qui était brésilien donc c'était plus compliqué. Et puis on a racheté les actifs d'une petite boîte qui s'appelait Lycoris, et là ça n'a pas très bien marché. »* (Ex-Directeur Général, Mandriva).

A partir de 2005, MandrakeSoft est devenu Mandriva, ce changement était notamment dû à la fusion de la société MandrakeSoft avec Conectiva proposant aussi une distribution Linux nommée Conectiva. De manière résumée l'histoire de Mandriva Linux peut-être schématisée de la manière présentée ci-dessous.

[83] Blog de Gael Duval (2005), Fondateur de Mandrake.
[84] Un fork est en fait un logiciel open source dérivé d'un autre du fait de divergences technologiques ou idéologiques.

Figure 16 : Schématisation historique de Mandriva Linux



2.3.2. Les acteurs de Mandriva.

A. La particularité de Mandriva : la double nature de ses membres.

Mandriva est une entreprise particulière puisqu'elle gère à la fois des salariés et des contributeurs bénévoles appelés *« Cookers »*.

Contrairement à ce que l'on pourrait penser pour ce type d'entreprise, Mandriva est organisée de manière fonctionnelle et son organisation interne est très classique. La direction de Mandriva est en effet assurée par quatre directeurs pour les trois grands services de l'entreprise : produits, finance et vente, plus le directeur général. En tout, il y a quatre niveaux hiérarchiques.

D'après l'ancien Directeur général de Mandriva, la fonction de la hiérarchie est : *« Comme dans toutes les sociétés, faire fonctionner les choses de façon ordonnée. Il n'y a rien de spécial. […] Est-ce qu'il y a une antinomie entre […] développer un logiciel libre et le fait d'avoir une hiérarchie ? Non, il n'y en a pas. »* (Ex-Directeur Général, Mandriva).

En plus de cette structure fonctionnelle, les salariés de Mandriva travaillant sur le développement sont très spécialisés : *« Il y a un mec qui fait KDE, il y a un mec qui fait GNOME, il y a un mec qui s'occupe que d'imprimantes, il y en a un qui s'occupe que de Kernel… »* (Ex-Directeur Général, Mandriva).

La direction générale essayait d'incorporer plus de polyvalence chez les développeurs, la solution envisagée est le changement de postes. L'aspect plus particulier de Mandriva est le fait que la société gère la contribution de bénévoles : *« Le produit inclut des parties que nous avons développées ainsi que des éléments disponibles déjà gratuits, que nous avons recueillis. Il est difficile d'évaluer la part produite par nous et celle que nous intégrons, mais on peut penser que cette part est aussi de l'ordre de 1,5 %. »* (Ex-Directeur Général, Mandriva).

Le cas Mandriva montre que le rôle des Cookers est essentiel dans le modèle économique de l'organisation puisqu'environ 98,5% du code des produits Mandriva n'est pas développé par la firme. Plus généralement, les firmes de l'industrie du logiciel libre ne peuvent plus raisonnablement être modélisées sous la forme d'une fonction de production.

Pour rendre compte de la diversité de la firme nous avons interrogé à la fois des membres des employés et des Cookers.

B. Regard sur quelques acteurs de Mandriva.

→ L'ancien Directeur général :

L'ancien Directeur général se décrit en ces termes : *« Ecoutez, j'ai 58 ans. J'ai passé la moitié de ma carrière dans la recherche et l'Université, la moitié dans l'industrie et puis j'ai passé un tiers aux Etats-Unis, deux tiers en France. J'ai pris ce boulot en revenant, donc je ne suis pas un fondateur… Ca fait trois ans et demi que je suis ici. […] Je viens de l'industrie du logiciel, mais je ne connaissais pas du tout le logiciel libre, je ne connaissais pas Linux. »* (Ex-Directeur Général, Mandriva).

Au moment où l'ex-Directeur général de Mandriva est arrivé la société était en difficultés, il décrit sa prise de fonction dans la société : *« Au début, ils m'ont assez mal reçu… Ils m'ont pris pour un mercenaire et un voyou… Euh, un mercenaire et un capitaliste. Et puis petit à petit, ils s'y sont habitués. Donc ils reconnaissent à peu près que je sais faire les choses… Mais il ne me considère pas comme un technicien, bien que je l'ai été à une époque, mais*

effectivement je ne le suis plus. Et, […] ils croient que je vais leur résoudre tous leurs problèmes quand même… A tort ou à raison… » […]

« J'ai eu le boulot puisque personne le voulait, c'est assez simple ! […] La société allait tellement mal que personne n'a repris le boulot. Moi je pensais qu'il y avait un actif extraordinaire, c'est-à-dire personne pensait, quand je l'ai reprise en […] 2002-2003, […] personne pensait qu'on pouvait réussir sans argent. Et moi j'ai regardé, et je pensais qu'il y avait une chance de la faire réussir sans argent, c'est comme ça que je l'ai eue. J'ai regardé un certain nombre de boîtes, et c'était certainement la plus « fun », et la plus risquée. » (Ex-Directeur Général, Mandriva).

Pour lui c'est évident : *« L'open source ça ne peut pas marcher que sur du bénévolat ! Ca, j'en suis convaincu. » […] D'abord, il faut des entreprises, sinon ça ne se diffuserait pas. […] Mais surtout, il faut un intérêt économique pour les gens, ça peut pas être le pur plaisir de contribuer à quelque chose, ce n'est pas vrai ça marchera pas ! Si on discute avec les gens de la fondation Apache, etc. c'est des gens frustrés, malheureux de ne pas gagner d'argent… Donc, c'est dur ! Même si c'est une réussite. C'est une superbe réussite. Il faut des récompenses. On vit dans une société qui est très mercantile et qui valorise l'argent, les profits… donc on échappe difficilement à ça quand même ! »* (Ex-Directeur Général, Mandriva).

→ **Ingénieur 1 :**

Ingénieur 1 est un salarié de Mandriva. Il se présente en ses propres termes : *« Je suis un ingénieur en informatique qui utilise linux depuis 5 ou 6 ans et je passe pas mal de mon temps libre à participer à divers projets liés au logiciel libre. […] Concernant la société Mandriva, j'y suis employé dans l'équipe service qui réalise des développements ou du conseil en rapport avec la distribution. Je suis responsable d'un projet pour le ministère de la défense. Concernant la distribution, je participe depuis 2001 au développement, je package des logiciels depuis 2002. Je participe aussi à des projets en rapport comme le PLF (qui distribue des logiciels supplémentaires) ou Youri qui est une infrastructure de gestion de distribution. Je fais partie, je pense, des 15 plus anciens contributeurs. »* (Ingénieur 1, Mandriva).

Ingénieur 1 explique ce que représente la distribution pour lui : *« Mandriva est le système que j'utilise tous les jours. L'améliorer me profite directement, et beaucoup de mes amis et de ma famille l'utilisent ou y participent. Toute ma vie actuelle est donc liée à ce projet. De plus maintenant que je suis salarié depuis 2 ans, l'amélioration de la qualité influe sur l'avenir de*

mon employeur donc sur mon emploi mais cela n'est pas devenu une réelle source de motivation. » (Ingénieur 1, Mandriva).

Ingénieur 1 a longtemps été contributeur avant d'être recruté au sein de Mandriva. Pour lui il s'agit réellement d'une convergence d'intérêts. Pour Mandriva cela constitue certainement un gain de temps en formation puisque sans faire partie de l'entreprise, le nouvel employé connaissait déjà le logiciel de manière pointue. Ingénieur 1 relate l'histoire de sa participation à Mandriva : *« J'ai commence à utiliser Mandriva a plein temps vers 2000 (à ne plus utiliser Windows donc) en septembre 2001 quand j'ai eu l'ADSL j'ai pu utiliser la version de développement et le mettre à jour quotidiennement à partir de ce moment la j'ai aussi pu signaler les bugs que je rencontrais. J'ai ensuite fait connaissance par internet avec les employés et les contributeurs et avec certains en vrai et courant 2002 on m'a proposé de corriger moi même les problèmes et de m'occuper moi même de mes logiciels vu que je commençais à bien connaitre la distribution et à faire des choses propres. Ensuite au fur et à mesure mes droits ont augmenté je pense que depuis début 2003, je fais partie des contributeurs de confiance qui ont a peu près tous les droits. »* (Ingénieur 1, Mandriva).

→ **Cooker 1.**

Cooker 1 se décrit lui-même comme : *« Un ingénieur en informatique actuellement en mission longue dans une administration, en tant qu'administrateur système et réseau. Je travaille dans une start-up spécialisée dans la sécurité et les logiciels libres depuis 1 an et demi. »* (Cooker 1, Contributeur bénévole, Mandriva).

Il poursuit en expliquant son rôle dans le développement de la distribution : *« Je m'occupe de certains composants du système, je participe aux tests des versions via le processus de cooker, j'essaye de discuter des choix techniques et des problèmes sur les listes de diffusions. Je passe aussi pas mal de temps sur le salon (et d'autres) à participer. » […] Mon rôle, en tant que mainteneur de paquet (le paquet étant un logiciel ou un composant) est d'intégrer un logiciel externe, fait par une tierce personne, de voir que la dernière version est disponible pour les utilisateurs, qu'elle marche, et qu'elle s'intègre bien, (par exemple, les menus, la documentation) [avec] les autres paquets. Je dois aussi gérer les rapports de bugs sur mes paquets, et j'en remplis quand j'en croise ailleurs, ou je corrige directement. J'interagis avec les autres développeurs, aussi bien de Mandriva, que de projets externes (python, kde, etc.).* (Cooker 1, Contributeur bénévole, Mandriva).

Pour lui, Mandriva Linux représente : *« Un moyen de bosser sur ce que j'aime, une façon d'améliorer ce que j'utilise et de contribuer à quelque chose d'utile pour tous. »* (Cooker 1, Contributeur bénévole, Mandriva).

Cooker 1 relate les circonstances qu'ils l'ont amené à contribuer à Mandriva Linux : *« Oui, j'ai commencé à utiliser linux quand je suis rentré en Fac. Je n'avais pas l'accès à internet chez moi, donc j'ai profité de la Fac pour chercher des informations sur la sécurité, comprendre finalement, si les pirates informatiques des films n'étaient que des exagérations ou pas, si les virus étaient si dangereux, et qui pouvaient avoir envie de faire ça. En cherchant, j'ai découvert le fonctionnement des réseaux, etc., et les systèmes unix. J'avais toujours voulu programmer et ce genre de système me semblait bien plus amusant que celui que j'avais. J'ai donc commencé à voir comment en obtenir un, par exemple, via un magazine. J'avais au début des problèmes, bien sûr, notamment du matériel mal reconnu donc j'alternais avec Windows pendant un temps. J'ai commencé à avoir des TP sur Unix, et l'expérience acquises au cours de ma scolarité, j'ai eu l'occasion de partir à Montréal cela m'a permis de comprendre un peu plus facilement. J'ai donc acheté un portable sur l'argent de mon boulot d'été, j'ai remis une Mandrake (l'ancien nom de Mandriva) et je suis parti. Mon binôme était aussi un utilisateur de linux, et était beaucoup plus impliqué que moi dans le domaine, et il m'a donné envie d'en faire plus. J'ai donc commencé à m'inscrire sur les listes de diffusions ayant trait au développement de Mandrake (la liste cooker) et j'ai fait des rapports de bugs après quelques mois, au cours de mon premier stage. Je me suis aussi impliqué dans le groupe d'utilisateur linux près de chez moi, ce qui m'a permis de décrocher mon deuxième stage. [..] C'est aussi à cette époque que j'ai commencé à traîner sur l'IRC et à faire des contacts avec d'autres développeurs. »* (Cooker 1, Contributeur bénévole, Mandriva).

Cooker 1 explique les raisons qui justifient sa participation au développement de Mandriva Linux : *« Pour plusieurs raisons, la communauté est relativement petite et à taille humaine, et je trouve ça mieux. Il n'y a pas une organisation bureaucratique lourde, comme Debian ou des projets plus gros et il y a une relative liberté d'action et une volonté de faire avancer les choses sans s'enliser dans des débats. Les gens sont sympas, et j'ai un feedback direct de mes contributions, surtout quand ça sert à résoudre des problèmes que je rencontre et j'ai commencé par ça, et je n'ai pas de raison de changer. »* (Cooker 1, Contributeur bénévole, Mandriva).

C. Le rapport entre contributeurs et salariés de Mandriva.

Le rapport entre contributeurs et salariés est particulièrement intéressant et laisse apparaître des éléments particuliers. D'après Ingénieur 1 : *« Les contributeurs sont assez variés, beaucoup sont administrateurs et participent car ils ont besoin des logiciels pour leur travail, d'autres sont étudiants et font ça pour s'amuser. En général ils n'ont aucune idée des contraintes "produit" et veulent surtout répondre à leurs besoins, mais les principaux contributeurs ont quand même envie que la distribution réussisse et que la société gagne de l'argent pour mettre plus de moyens sur la distribution. Les employés ont aussi des profils variés, pour beaucoup de ceux qui participent à la distribution c'est leur premier emploi et ils font ça parce que ça leur plait et ne sont pas forcement plus intéressés par les problématiques commerciales. Certains se jugent "au dessus" des contributeurs et refusent de collaborer mais la plupart voient les contributeurs comme une ressource très utile ou sont eux-mêmes d'anciens contributeurs donc les traitent sur un pied d'égalité. Globalement tout le monde essaye d'améliorer la distribution selon ses besoins et même si certains refusent de collaborer et sont virulents sur les listes de diffusion, tout le monde arrive à travailler ensemble car quelques personnes ont une réelle autorité. Certains contributeurs considèrent (à juste titre en général) qu'ils ont fait tellement pour la distribution qu'ils mériteraient plus de considération et les relations entre les "vieux" contributeurs et certains employés est parfois difficile mais le problème se pose aussi dans des entreprises (et en particulier chez Mandriva ou il y a pas mal de tensions internes) »* (Ingénieur 1, Mandriva).

Ingénieur 1 met en évidence un phénomène tout à fait particulier puisqu'il explique que certaines personnes sont reconnues et respectées à la fois à l'intérieur et chose plus surprenante à l'extérieur de l'entreprise. Nous sommes donc bien loin du discours consistant à dire qu'il n'y a pas de hiérarchie dans les communautés open source. Ce témoignage en est un exemple saillant. Cooker 1 souligne également la présence de ce qu'il nomme *« une organisation bureaucratique lourde*[85] *»* dans certaines communautés de taille plus importante. Ensuite, le passage du statut de contributeur à salarié de l'entreprise est un phénomène également singulier.

[85] Cooker 1, Contributeur bénévole, Mandriva.

2.3.3. Les systèmes de récompenses et le financement de Mandriva.

Dans certains cas, les Cookers de Mandriva Linux sont récompensés de manière informelle. D'après l'ex-Directeur général : *« Il y a [des systèmes de récompenses] […] informels, il est arrivé qu'on file des trucs, qu'on envoie des cadeaux. Déjà, il y a le système de les inclure dans le club. Ils sont membres du club automatiquement [Club payant de la société donnant accès à un certain nombre d'éléments particuliers comme de la documentation]. »* (ex-Directeur général, Mandriva).

Mandriva est une société anonyme à conseil d'administration. Elle a un mode de financement très classique puisque : *« La société est cotée au marché libre, donc il y a 1600 actionnaires à peu près. Les dix premiers doivent contrôler ensemble 60% du capital. Donc le premier actionnaire c'est Millenium Partners qui est un edge fund new-yorkais, le deuxième c'est… Remote Reward qui est un fonds français, en troisième il doit y avoir les fondateurs, donc il y a des fonds et des personnes physiques. »* (ex-Directeur général, Mandriva).

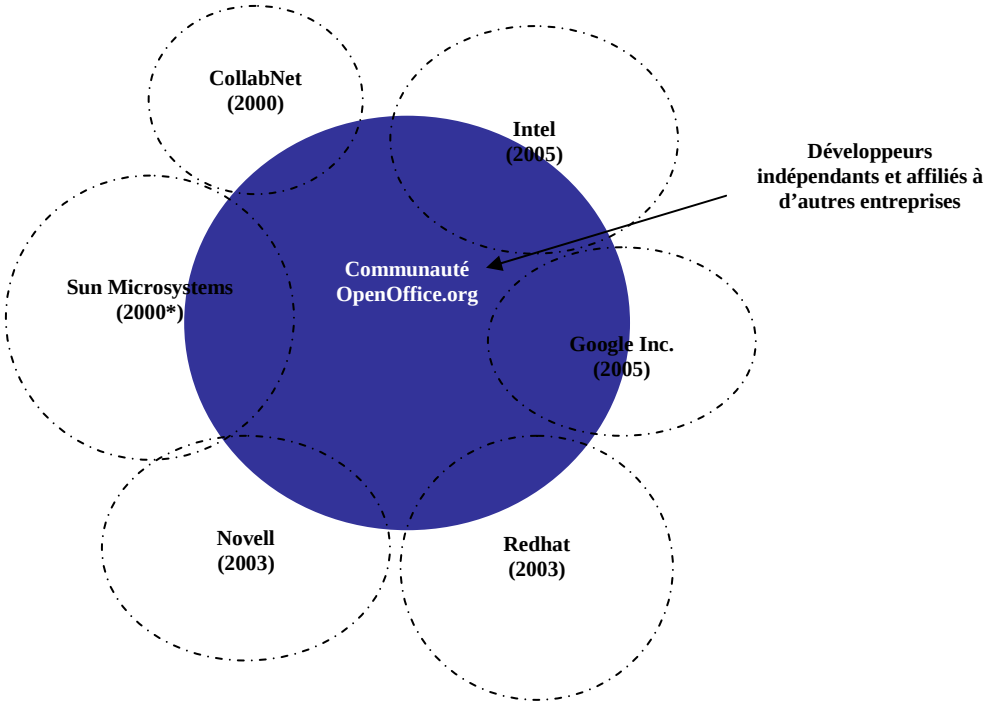
2.4. Le cas OpenOffice.org : une communauté interentreprises.

2.4.1. Vue générale d'OpenOffice.org

Dans OpenOffice.org *« La majorité des contributeurs font partie de Sun, Novell, Redhat, Intel et Google. Il y a peu de développeurs indépendants. »* (Membre du Community Council 1, OpenOffice.org).

Etant donné qu'OpenOffice.org est composé principalement d'employés de grandes entreprises, nous nous intéresserons aux enjeux rapportés au niveau des entreprises impliquées dans OpenOffice.org. Il convient de souligner que le groupe IBM a officiellement déclaré en 2007 qu'il participerait à OpenOffice.org (OpenOffice.org 2007). Bien que lors de notre premier entretien IBM refusait de participer principalement à cause du fait que Sun détenait (et détient toujours) le copyright d'OpenOffice.org. Toutefois, la participation d'IBM est faible (en 2009) d'après les autres membres de la communauté. Le schéma ci-dessous synthétise les entreprises impliquées dans OpenOffice.org.

Figure 17 : Les entreprises impliquées dans OpenOffice.org



2.4.2. Gouvernance et régulation dans OpenOffice.org

A. La gouvernance d'OpenOffice.org.

→ Le Community Council.

La direction de la communauté OOo est assurée par le *Community Council* (CC). Le CC supervise l'ensemble du développement d'OOo et joue le rôle d'un comité de pilotage stratégique. Il est composé de dix membres élus dans les proportions exposées ci-dessous. Dix représentants : (1) trois représentants des contributeurs en code ; (2) trois représentants de l'ensemble des contributeurs à OOo en documentation, marketing, etc. ; (3) deux représentants des projets Native-Lang (projets linguistiques) ; (4) un représentant des utilisateurs ; (5) un représentant de Sun Microsystems. Actuellement le CC comprend trois employés de Sun Microsystems, un employé de CollabNet et cinq membres indépendants.

→ L'Engineering Steering Committee.

Il existe un second organe de direction dans OOo, celui-ci est technique. Il s'agit de l'Engineering Steering Committee (ESC). Le rôle de l'ESC est de réaliser des recommandations en matière d'implémentation technique en cas de conflit d'intérêts. La décision de l'ESC peut-être invalidée par le Community Council.

La condition pour faire partie de l'ESC est la suivante : avoir contribué activement des correctifs à OOo durant 2 ans et dans plus d'un projet. Il y a quatre types de membres dans l'ESC : (1) les représentants des contributeurs ; (2) les chefs de projet ; (3) des membres du Community Council qualifiés de *« senior developer »* ; (4) des membres invités. L'ensemble des membres ont un droit de vote sauf les membres invités. Depuis 2008, la composition de l'ESC est la suivante : six membres permanents : dont deux employés de Sun Microsystems, un employé de Novell, un employé de Red Hat et deux indépendants ; quatre membres invités : trois représentants de distributions Linux : Redflag, Ubuntu, Debian et un représentant d'IBM ; quatre chefs de projets tous de Sun Microsystems.

B. Les mécanismes de régulation des droits d'auteurs.

Le code source et la marque OpenOffice.org sont la propriété de Sun Microsystems. Les développeurs participant substantiellement au développement d'OOo ont signé un contrat de copyright conjoint. Le SCA (Sun Copyright Assignment) est un contrat de droit privé permet le partage de copyright entre une personne participant au développement d'OOo et Sun Microsystems. Ce document confère à Sun Microsystems les mêmes droits que l'auteur du code contribué à OpenOffice.org. Il n'est pas possible de participer au développement d'OOo (en matière de code) sans signer ce contrat. En date du 3 juin 2008, 880 personnes avaient signé un SCA avec Sun Microsystems.

Comme nous l'avons précisé plus haut, la communauté OpenOffice.org est constituée principalement d'entreprises. Ces acteurs institutionnels seront présentés sous deux angles : les enjeux de leur participation et leur rôle respectif dans OpenOffice.org.

2.4.3. Les membres d'OpenOffice.org : enjeux et rôles respectifs.

A. Sun Microsystems Inc.

La libération de StarOffice par Sun Microsystems peut être justifiée principalement par deux raisons. Premièrement, au moment de la libération de StarOffice, le marché de la bureautique était dominé par Microsoft Office. Avec cette stratégie, Sun souhaite imposer des standards ouverts dans la bureautique pour pouvoir améliorer l'interopérabilité des systèmes. A plus long terme, Sun Microsystems envisage de concurrencer Microsoft sur le marché des systèmes d'exploitation. Les formats de documents de Microsoft Office se sont, en l'espace d'une dizaine d'années, imposés *de facto* à la fois dans les mondes de l'entreprise et de l'informatique grand public.

Deuxièmement, les développements de StarOffice et d'OpenOffice.org sont menés conjointement depuis environ 2004. D'après un représentant de Sun : *« Il n'y a pas les développeurs StarOffice. Aujourd'hui il y a une équipe qui est, pour l'essentiel à Hambourg qui développe OpenOffice.org, et donc leur repository se trouve sur OpenOffice.org et il y a d'autres membres qui font StarOffice et qui utilisent le noyau OpenOffice.org pour fournir StarOffice. »* (Technology Advisor 1, Sun Microsystems).

L'ouverture du code source de StarOffice a permis à Sun de partager le coût de développement de StarOffice avec d'autres entreprises. StarOffice est une version d'OpenOffice.org commercialisée par Sun. StarOffice propose un certain nombre de fonctionnalités en plus par rapport à OOo en revanche : *« C'est exactement le même logiciel on rajoute des polices, on rajoute une galerie, etc. Tout ce que nous faisons c'est rajouter des choses autour d'OpenOffice.org. Le code, il n'y a pas une virgule de différent. »* (Technology Advisor 1, Sun Microsystems).

Le rôle joué par Sun dans le développement d'OpenOffice.org (OOo) est essentiel : Sun est à la fois le fondateur d'OOo et son principal financeur. Les employés de Sun sont présents à tous les niveaux d'OOo. (1) Du point de vue de la direction opérationnelle, un nombre important de lead et co-lead sont employés par Sun. En revanche, il y a des différences en ce qui concerne le type de *« projets*[86] *»* menés. Par exemple, plus de sept *« Accepted Projects »* sur dix sont dirigés par des employés de Sun, ce rapport tombe à environ cinq sur dix en ce qui concerne le poste de co-lead dans cette même catégorie. Dans la catégorie *« Incubator »* le rapport tombe à un peu plus d'un sur dix. Pour la catégorie *« Native-Lang »*, la concentration de développeurs indépendants est la plus forte. Au vue de cette présence, nous pouvons dire que les employés de Sun sont placés à des niveaux permettant à la firme d'avoir un contrôle relatif de l'évolution du cœur d'OpenOffice.org.

(2) Du point de vue technique, Sun est également représenté par six personnes sur quatorze membres à l'Engineering Steering Committee (ESC).

(3) Du point de vue stratégique, la présence de Sun est encore plus forte, sur les neuf membres du Community Council, trois sont employés de Sun et un est employé par CollabNet (la société mandatée par Sun pour superviser le développement d'OOo). Les employés de Sun ont un rôle clé dans le développement d'OOo, certains membres de la communauté étaient déjà employés par StarDivision. Ces quelques acteurs ont un rôle

[86] Terminologie d'OOo.

important dans la mesure où ces derniers ont une connaissance globale du code source d'OOo représentant plus de 8 millions de lignes de code.

B. Intel Corp.

La participation d'Intel au développement d'OOo est, au premier abord, inattendue. Pour le moins, si l'on ne prend uniquement en compte le métier d'Intel. Mais en analysant de plus finement la participation d'Intel à OOo celle-ci n'est plus aussi surprenante. En effet, Intel est un fabricant de processeurs, il a donc intérêt à ce que ses puces soient utilisées à hauteur de leur capacité. Pour qu'un logiciel ait un fonctionnement optimal dans un environnement celui-ci doit être conçu en prenant en compte la puce qui recevra les instructions du programme. Intel a donc intérêt à ce que la suite bureautique OOo exploite au mieux la puissance des processeurs Intel : ce qui en retour, permet à Intel de justifier l'utilisation de ses microprocesseurs[87]. En outre, il convient de noter qu'Intel a subi de fortes pressions de la part de Microsoft. En 1999, Intel fut contraint d'arrêter le développement de logiciels assurant l'interopérabilité de ses processeurs (Jackson 1999: 19).

La participation d'Intel se situe au niveau de la recherche de compatibilité totale d'OOo avec les plateformes Intel[88]. Intel participe notamment au développement des performances d'OOo pour Linux. Cet engagement fait écho à une volonté plus générale de la part d'Intel d'investir dans la promotion de Linux et d'autres logiciels open source. Intel contribue au développement d'OOo depuis environ 2005. La participation d'Intel à OOo s'est déroulée en plusieurs temps.

Premièrement, de janvier à février 2005, les employés d'Intel ont réalisé (1) une *« Gap Analysis »* cela signifie qu'ils ont comparé les performances attendues et réalisées d'OOo ; (2) une analyse des caractéristiques d'OOo ; (3) une analyse de l'interopérabilité d'OOo. Deuxièmement, de mars à avril 2005, ils ont (1) observé la construction d'OOo ; (2) lu le code source ; (3) étudié les manuels développeurs d'OOo. Suite à cette phase, trois grands domaines d'intervention ont été identifiés : (a) le démarrage (*« Start-up »)* car les développeurs d'Intel ont remarqué que des choses pouvaient être améliorées ; (b) la conversion des instructions d'OOo par le processeur *(« Rendering »)* car il y avait le potentiel d'améliorer les performances d'OOo, notamment en ce qui concerne le graphisme ; (c) le

[87] Pour de plus amples informations sur la stratégie d'Intel en matière de coopération voir les travaux de Gawer (2000).
[88] L'analyse du rôle d'Intel dans le développement d'OOo est largement basée sur la présentation de Keskar D. & Leibowitz M., *Speeding OpenOffice Startup - Profiling, Tools & Approaches*, OpenOffice Conferences, 29 septembre 2005.

processus de division de tâches en sous processus (*« Threading »)* parce qu'Intel désirait faire profiter OOo de la technologie *« Hyper-Threading*[89] *»*. Troisièmement, de mai à juin 2005, ils ont reçu l'approbation pour le projet et ont proposé un cadrage initial. Quatrièmement, en juillet 2005, ils ont proposé des patchs. Cinquièmement, ils ont utilisé des outils, réalisé des analyses de démarrage. C'est à ce moment que l'équipe d'Intel dédiée à OOo a grossi : de deux à dix personnes.

Grâce à l'intervention d'Intel, la version 2.0 est plus rapide que la version 1.1. Pourtant la version 2.0 offre beaucoup plus de fonctionnalités. Il convient de noter que le rôle qu'Intel n'était pas connu à l'avance. Cette incertitude se trouvait aussi bien au niveau du rôle que des ressources. Concernant le rôle d'Intel, il y a bel et bien eu une *étude exploratoire* qui a permis de mettre en évidence des champs d'action pour Intel, ensuite trois champs d'action ont été sélectionnés. En ce qui concerne les ressources allouées, l'incertitude était également importante : au départ il n'y avait que deux personnes d'Intel chargées d'analyser OOo et suite à l'investigation réalisée l'équipe fut renforcée.

C. Google Inc.

Google est une société relativement jeune même si elle a aujourd'hui un rôle clé dans l'industrie informatique. En 2005, Google lança les *« Google Summer of Code »*, une initiative visant à payer des étudiants pendant les vacances pour participer au développement de logiciels open source. La session 2005 de l'opération a concerné 400 étudiants répartis dans 40 organisations de l'open source. D'après Chris DiBona, la participation de Google au développement de logiciels open source se justifie par le fait que Google *« croit que l'open source représente un des moyens les plus sûrs de préserver la concurrence dans l'industrie informatique*[90] *»*.

La participation de Google à OOo a officiellement été mentionnée dans un accord entre Google Inc. et Sun Microsystems le 4 octobre 2005. D'une part, Sun Microsystems s'est engagé à proposer la barre d'outils de Google (*« la Google Toolbar »*) en tant qu'option de son Java Runtime Environment (JRE) et d'autre part, Google s'est engagé à participer au développement d'OOo, en revanche aucun objectif quantitatif ou qualitatif n'a été donné pour

[89] Cette technologie consiste à découper des instructions machine en de multiples flux permettant d'améliorer le temps de traitement de celles-ci en agissant sur la capacité de traitement et non directement sur la vitesse de traitement, le gain est donc indirect.
[90] Propos recueillis auprès de DiBona C., *Open Source Programs Manager*, Google Inc., le 11 juillet 2006.

cette participation. Ensuite, nous savons que Google aide au développement d'OOo en finançant des étudiants à travers les *Google Summer of Code*.

Il convient également de souligner que Google a adopté le format de document ODF (Open Document Format) de la suite bureautique OpenOffice.org dans ses différents services en ligne. D'après un représentant de Sun Microsystems : *« le prochain enjeu sera plus dans le format de données et les services que l'on va fournir autour que le logiciel qui lui sur un poste de travail va être capable de pouvoir éditer un texte. Et le véritable enjeu est là. C'est ici que Microsoft a des problèmes avec OpenXML à cause du fait qu'il n'est pas standardisé*[91] *que Google fasse massivement de l'ODF, pour eux pourrait être quelque chose d'extrêmement grave. Tous les nouveaux services et inéluctablement on fera de la bureautique sur internet pourrait leur échapper. »* (Technology Advisor 1, Sun Microsystems).

Très récemment, la participation de Google dans OOo a largement été réduite. D'après Eric Bachard (OpenOffice.org), « l'investissement de Google, depuis plus d'un an, me semble moindre. Pour illustrer mon propos, cela fait maintenant 2 ans que le projet OpenOffice.org n'a plus de slots pour le Summer of Code, et la réponse « politiquement correcte » qui m'a été faite par exemple par Leslie Hawthorn [Google Inc.], c'est que « tous les projets ne peuvent être retenus ». En vérité, le torchon brule entre Google et Sun, mais je ne souhaite pas aller sur ce terrain, car je n'ai pas assez d'informations. » (Séguin 2009).

D. Red Hat Inc.

La participation de Red Hat à OOo n'est pas surprenante, elle procure à Red Hat un élément de différenciation supplémentaire face aux autres distributions Linux. Red Hat propose différentes versions de son système d'exploitation avec OOo intégré. Sun Microsystems a des accords commerciaux avec Redhat.

La participation de Red Hat se situe essentiellement dans la création de filtres Writer. Cette fonctionnalité consiste à assurer l'importation et l'exportation de documents provenant d'autres suites bureautiques (MS Word, WordPerfect, etc.). La participation de Red Hat a été facilitée grâce à la présence d'un ancien employé de Sun Microsystems, qui travaillait déjà sur la suite bureautique.

[91] Au moment de notre entretien le format OpenXML n'était pas encore certifié ISO.

E. Novell Inc.

Les raisons justifiant la participation de Novell à OOo sont du même ordre que ceux de Red Hat. En outre, Novell a également des accords commerciaux avec Sun Microsystems.

Novell[92] participe au développement d'OOo en allouant des développeurs. Novell concourt au développement d'OOo en essayant de maximiser ses intérêts. Ainsi, Novell finance et assure le développement d'une version d'OOo pour KDE. KDE est un environnement de travail graphique disponible pour différents systèmes d'exploitation (Linux, FreeBSD, Sun Solaris, etc.). La direction du développement d'OOo pour KDE est assurée par un employé de Novell. La participation de Novell au développement d'OOo pour KDE se justifie par le fait qu'OOo est la suite bureautique la plus complète et performante existante pour l'environnement KDE. Par exemple, KOffice (une suite bureautique concurrente pour KDE) ne propose que très peu de fonctionnalités comparé à OOo ou MS Office.

Novell participe également à la réalisation d'un module d'import et d'export du format OpenXML. Il convient de souligner que Novell a signé un accord avec Microsoft prévoyant une coopération technique portant notamment sur le développement d'un module visant à améliorer l'interopérabilité entre Microsoft Office et OpenOffice.org (Novell Inc. 2007: 24). Le problème est que la spécification du format OpenXML, le format ouvert de la suite Microsoft Office 2007, est particulièrement complexe à implémenter. D'après Flguiere (Novell), le détail des spécifications du format OpenXML tient sur 5946 pages : il est par conséquent très long d'implémenter ce format (FIguiere 2007).

2.5. Le cas OW2 Consortium ou un exemple de consortium industriel.

2.5.1. L'histoire d'OW2 Consortium.

OW2 Consortium (OW2) est un regroupement d'entreprises et d'acteurs de la recherche ayant pour objet le middleware ou intergiciel. Le middleware est un *« domaine technique qui a pour vocation l'échange de données entre machines, applications et personnes. »* (Laisné 2006: 2). OW2 est en fait issu de la fusion de deux consortiums initialement distincts mais travaillant tous deux sur les problématiques du middleware open source : ObjectWeb et OrientWare.

[92] Lorsque nous parlons de Novell, nous incluons également SuZE qui fait désormais partie de Novell.

ObjectWeb a été fondé en 1999 par Bull (intégrateur), France Telecom (télécommunications) et l'INRIA (Institut National de Recherche en Informatique et Automatique). Au départ, le consortium était uniquement français et plus particulièrement localisé dans la région Rhône-Alpes. Puis, le consortium a gagné en notoriété et en taille jusqu'à devenir européen. ObjectWeb est issu de la convergence de trois briques technologiques : (1) JOnAS, (2) Jonathan et (3) Joram.

(1) JOnAS (Java Open Application Server) est un serveur d'application open source initié par Bull, une implémentation libre des spécifications J2EE (Java Entreprise Edition). JOnAS a été certifié par Sun Microsystems ce qui lui confère une crédibilité industrielle importante.

(2) Jonathan est un ORB (Object Request Broker) ou bus logiciel basé sur la norme CORBA (Common Object Request Broker Architecture) développé par France Telecom R&D (anciennement CNET).

(3) Joram est un système de messagerie permettant de répondre au problème de disparité technologique dans le cadre de systèmes d'informations distribués. De plus en plus d'entreprises rencontraient des problèmes d'interopérabilité entre des systèmes hétérogènes. Plus précisément *« la base technologique [de Joram] est une plateforme à agents distribuée, développée dans un centre de recherche commun [à] Bull, [l']Inria et les Universités de Grenoble. Lorsque la norme JMS est sortie, nous avons réalisé une implantation sur notre base technologique*[93]*. »* Joram a ensuite donné naissance à ScalAgent Distributed Technologies, une société de services assurant les activités d'édition de Joram. A la base, Joram avait été réalisé comme une alternative à des solutions non libres comme IBM WebSphere.

OrientWare[94] est un consortium issu des travaux initiés par le projet 863 chinois portant sur le middleware. OrientWare est le résultat de plus de 10 années de recherche et développement financés par le gouvernement chinois. Les cinq premières années (1996-2000) ont permis la création de plusieurs projets middleware comme un MOM (Message Oriented Middleware). Les cinq années suivantes (2001-2005) avaient pour objectif de faire du Middleware une activité indépendante.

OrientWare a été initié en octobre 2004 par des acteurs de la recherche : BHU (Beihang University), ISCAS (Institute of Software, Chinese Academy of Science), NUDT (National

[93] Serge Lacourte, Directeur général, ScalAgent Distributed Technologies, entretien en ligne avec l'auteur, 17 juillet 2009.

[94] La description d'OrientWare est basée sur deux présentations réalisées d'une part par W. Huaimin (NUDT), 2006, Orientware and OW2 Cooperation : Leading the world by establishing a mutually beneficial cooperation et d'autre part par H. Mei (PKU), 2005, Middleware R&D in 863 High-Tech Program.

University of Defense Technology) et PKU (Peking University), et des acteurs de l'industrie : Intervision Software Corp., TongTech et CVIC SE.

OrientWare est à l'origine de différentes briques logicielles comme StarAppServer un serveur d'application basé sur CORBA, PKUAS un serveur d'application basé sur J2EE. Les composants développés dans le cadre d'OrientWare ont été employés dans des contextes industriels variés comme les télécommunications, le transport et la gestion du trafic aérien. Ces applications montrent la crédibilité et la haute fiabilité des composants développés.

2.5.2. Une vue générale d'OW2.

OW2[95] est un consortium visant à développer une base de code middleware open source mais également à créer des opportunités économiques pour ses participants ce que Cédric Thomas, CEO d'OW2, nomme un *« écosystème d'affaires »* (Thomas 2007: 6).

OW2 a trois objectifs stratégiques : *« (1) établir un leadership sur le marché du middleware en général à travers du code open source ; (2) construire une organisation avec des ressources financières pour agir en autonomie ; (3) fournir une plate-forme où des logiciels open source au niveau mondial. »* (Thomas 2007: 7).

OW2 est un consortium composé de trois éléments : (1) une base de code issue de la convergence d'ObjectWeb et d'OrientWare ; (2) une communauté développant les logiciels et travaillant également sur ce que les membres d'OW2 appellent des activités ; (3) un système de gouvernance conférant à OW2 une relative indépendance.

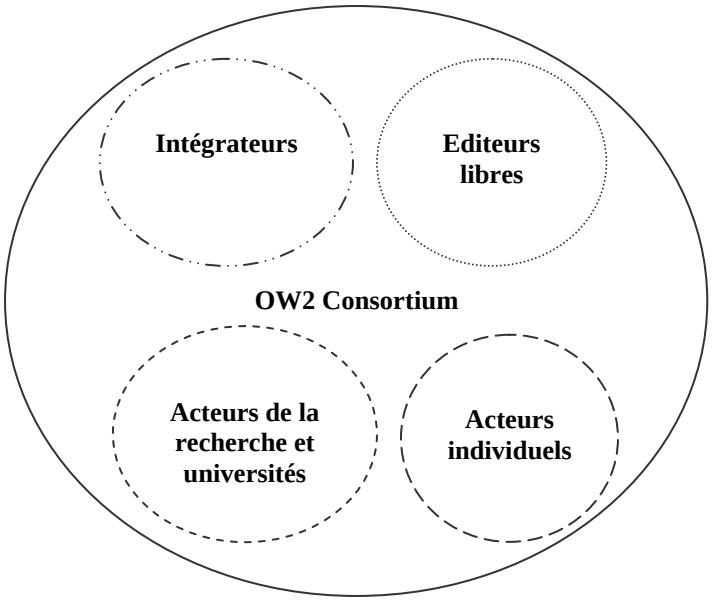
L'élément particulier du consortium OW2 est l'affichage clair et sans complexes de la dimension affaires contrairement à beaucoup d'autres communautés. La dimension affaires n'est pas seulement affichée, c'est aussi un but à part entière que les membres de la communauté OW2 mettent en avant. Dans les trois activités principales du Consortium, il y a une catégorie d'activités visant particulièrement à développer des affaires en combinant des logiciels existants dans la base de code OW2 ainsi que d'autres logiciels open source.

Il y a quatre catégories[96] d'organisations impliquées dans OW2 : (1) les intégrateurs, (2) les éditeurs, (3) les acteurs de la recherche et de l'enseignement et (4) les individus.

[95] Cette partie est basée sur les propos de Cédric Thomas, CEO, OW2 Consortium, Keynote OW2, 25 octobre 2008 et Cedric Thomas, Leading Open Source Middleware : OW2 General Presentation, Janvier 2007.

[96] Bien que Cédric Thomas propose neuf catégories, nous pensons qu'il est possible de regrouper certaines catégories en fonction des caractéristiques et des intérêts motivant la participation à OW2. C'est pourquoi nous pensons qu'il y a quatre catégories de membres. Toutefois, nous avons utilisé les descriptions proposées par C. Thomas (2009 : 10-11).

Figure 18 : Structure simplifiée d'OW2.



(1) La catégorie des intégrateurs doit être considérée au sens large : elle comprend un ensemble d'entreprises de taille importante (Bull) voir très importante (Thales). Ces entreprises utilisent l'open source comme des briques technologiques afin de développer des solutions sur-mesure. Pour ces firmes, le middleware ne constitue pas un élément de différenciation : il s'agit toutefois d'un enjeu important en termes d'indépendance technologique surtout pour certains projets. D'après le Directeur de la Recherche de l'Innovation de Thales D3S, « *Le middleware […] est utilisé par les grands groupes comme BULL, THALES, FT [France Telecom] pour construire des applications, services et/ou solutions qui sont elles vendus aux clients finaux. L'utilisation de l'OSS permet ici de baisser le cout général d'acquisition par rapport à des solutions propriétaires clientes (IBM par exemple)*[97]*. »*

(2) La catégorie des éditeurs open source comprend des entreprises de taille réduite ayant développé un middleware libre ainsi qu'une expertise pointue associée, *« c'est sur cette expertise [que ces firmes] basent leur légitimité dans le consortium OW2*[98]*. »* Les éditeurs jouent un rôle particulièrement fondamental car ils font ce que les intégrateurs ne souhaitent pas faire. D'un autre côté, les éditeurs ne sont pas dimensionnés pour faire ce que les intégrateurs font. Les éditeurs et intégrateurs sont de ce point de vue complémentaires. Dans certains cas, les intégrateurs et les éditeurs s'associent pour présenter leurs complémentarités, c'est notamment le cas d'Exo Platform et Bull.

[97] Serge Druais, Directeur de la Recherche et de l'Innovation, Thales D3S Shanghai, entretien en ligne avec l'auteur, mercredi 4 aout 2009.
[98] Ibid.

(3) La catégorie des acteurs de la recherche et de l'enseignement contient des organisations ayant de forts liens avec l'industrie et recherchant à trouver des financements pour des projets de recherche. Certains acteurs comme l'INRIA en France ont déjà un statut académique et industriel reconnu internationalement. L'INRIA est d'ailleurs un des membres fondateurs d'ObjectWeb et continue de jouer un rôle important dans OW2. Pour les acteurs de l'enseignement, le logiciel libre est une opportunité importante puisque celui-ci permet de montrer concrètement à des étudiants comment un logiciel de qualité industrielle fonctionne.

(4) La catégorie des acteurs individuels comprend des individus travaillant essentiellement à leur compte en tant que consultant. C'est notamment le cas de l'ancien directeur exécutif d'OW2 désormais membre du comité de direction d'OW2 et consultant indépendant en technologies open source.

2.5.3. La gouvernance d'OW2.

Dans OW2, il y a trois niveaux de participation : Strategic member, Corporate Member et Individual. (1) Les *« Strategic Members »* sont des organisations souhaitant avoir une influence significative sur les orientations du consortium OW2. Toutefois, être Strategic Member (SM) signifie aussi investir de manière conséquente dans le développement du consortium. En outre, ils doivent conserver leur participation au minimum pendant trois ans, s'acquitter d'un droit de participation de 30 000 euros par an. De plus, ils doivent nommer un Directeur dans l'équipe de direction et un représentant dans chaque conseil, participer à une initiative, avoir au moins deux contributeurs dans un projet et fournir deux personnes à plein temps au Management Office (ou verser l'équivalent en numéraire 100 000 euros par personne). En 2009, cette catégorie contenait les 15 membres ci-dessous.

Tableau 17 : Strategic Members d'OW2 (2009).

Membres	Nationalité	Type
Alcatel Lucent	France	Industrie
Beijing University of Aeronautics & Astronautics	Chine	Recherche/Université
BULL SAS	France	Industrie
CVIC Software Engineering Co. Ltd.	Chine	Industrie
DOCSC	Chine	Industrie
Engineering Ingegneria Informatica S.p.A.	Italie	Industrie
France Telecom Group	France	Industrie
INRIA	France	Recherche/Université
Institute of Software, Chinese Academy of Sciences	Chine	Recherche/Université
National University of Defense Technology	Chine	Recherche/Université
NTCU	Chine	Recherche/Université
Peking University	Chine	Recherche/Université
Red Hat	USA	Industrie
SERPRO	Brésil	Industrie
Thales	France	Industrie

Concernant la nature des membres, plusieurs remarques peuvent être réalisées. Tout d'abord, les groupes industriels français ont un rôle central puisqu'ils représentent près d'un tiers des participants. Il ne faut toutefois pas oublier que le consortium OW2 est dans une large partie l'héritier d'ObjectWeb, une initiative française. D'autre part, les acteurs chinois ont également une place très importante car ils représentent près de la moitié des SM. La participation d'un groupe Brésilien montre à quel point le middleware est stratégique pour l'indépendance technologique d'industries émergentes.

(2) Les *« Corporate Members »* sont des organisations souhaitant participer à OW2 sans toutefois avoir suffisamment d'intérêts et/ou de ressources pour avoir un poids effectif sur la direction du consortium. Ces organisations doivent s'acquitter d'un droit de participation proportionnel à la taille de l'organisation (entre 1 000 et 6 000 euros).

(3) Les *« Individual Members »* sont des individus souhaitant participer à OW2 en leur propre nom car ces derniers ont développé une activité de conseil ou pour explorer l'opportunité d'une participation plus active à OW2 pour le compte d'une organisation. Cette catégorie ne paie pas de droits de participation.

Dans OW2, le *free-riding* est traité de manière intéressante : les participants à OW2 doivent contribuer suffisamment s'ils souhaitent avoir un poids effectif dans les orientations du consortium. Comme le souligne un représentant de Thales au board d'OW2, *« Il y a trois niveaux de contribution : strategic, corporate et individuel. Plus on paye et plus on peut influencer les choix. Au niveau strategic cela représente 30 000 euros par an. Pour Thales ce*

n'est pas grand-chose mais pour PME c'est non négligeable[99]*. »* Cette structuration des membres permet au consortium de créer un équilibre entre la contribution et les bénéfices attendus en termes d'orientation. En revanche, il est très difficile pour l'ensemble des entreprises interrogées de mesurer l'impact économique de leur participation à OW2.

OW2 dispose de trois types de collectifs dédiés à la gestion de l'organisation : un comité de direction (ou Board of Directors), un bureau de management (ou Management Office) et des conseils (ou Councils).

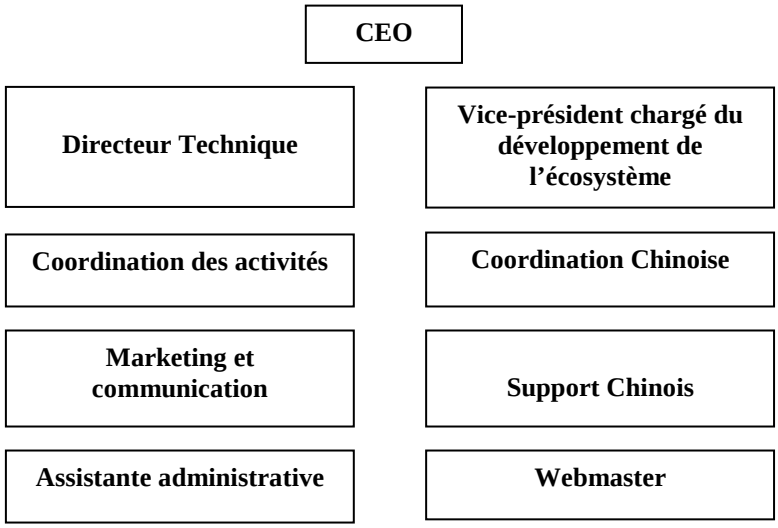
(1) Le comité de direction (*« Board of Directors »*) est constitué de l'ensemble des représentants des Strategic Members (14 représentants), des Corporate Members (14 représentants) et des participants individus (1 représentant). A travers la composition du board, nous voyons bien que les contributeurs individuels ont peu de poids dans les choix et les orientations stratégiques du consortium. En septembre 2008, Alcatel-Lucent a rejoint le consortium en tant que Strategic Member. Alcatel-Lucent était sur le point de publier une plateforme de services pour des applications en réseaux basée sur Java. La participation d'Alcatel-Lucent est clairement dans le but de disséminer ses technologies dans toute l'industrie grâce à l'envergure du consortium OW2.

Le Board of Directors traite des questions concernant sa propre composition, des conditions de participation dans le consortium et de tout ce qui est considéré comme relevant du niveau *stratégique* pour le consortium. Le comité de direction s'occupe également des questions légales comme par exemple le cas où un partenaire du consortium disparaitrai ou quitterai le consortium. En mars 2008, le board a décidé du rejet de la GPLv3 du fait de son incompatibilité avec la politique de propriété intellectuelle promue par OW2. En effet, OW2 donne la possibilité à ses participants d'utiliser du code breveté dans un logiciel libre sans pour autant perdre le bénéfice de sa protection intellectuelle.

(2) Le bureau de management (*« Management Office »*) a été créé suite à la croissance d'OW2 nécessitant une équipe chargée de faire fonctionner le consortium au quotidien. Ce bureau a à sa tête le CEO du consortium et 8 autres personnes (voir ci-dessous).

[99] Serge Druais, Directeur de la Recherche et de l'Innovation, Thales D3S, entretien téléphonique avec l'auteur, 5 aout 2009.

Figure 19 : Le Management Office d'OW2.



Historiquement, l'INRIA était *l'hébergeur* du consortium ObjectWeb jusqu'à la création d'une entité juridique autonome (association de loi 1901). La création du bureau de management a sans doute (en partie) été réalisée afin de combler la réduction du support de l'INRIA et la montée en charge d'OW2.

En outre, des fonctions spécifiques ont été créées pour la coordination avec les partenaires chinois d'OrientWare. En effet, des difficultés de communication ont été rencontrées avec certains partenaires comme DOCSC (une firme chinoise). Fin 2007, la firme était dans une phase de découverte de l'open source et la participation à OW2 était inexistante. De plus, des difficultés linguistiques se sont ajoutées. En général les entreprises chinoises contribuaient peu en 2007 car l'open source était nouveau sur le marché chinois.

(3) Il existe trois types de conseils (*« Council »*) : le conseil technologique, le conseil d'écosystème et le conseil opérationnel. (a) Le conseil technologique (*« Technology Council »*) a pour objectif de définir les orientations technologiques du consortium. Par exemple, le Technology Council (TC) définit la position du consortium en ce qui concerne les standards. Le TC réalise également des arbitrages pour savoir si un projet concurrence ou pas un autre dans le cadre d'une intégration dans OW2. Dans certains cas, un logiciel peut sembler concurrencer un autre d'un point de vue fonctionnel mais pas du point de vue technique, dans d'autres cas les projets sont complémentaires et les équipes travaillent ensemble pour exploiter au mieux ces complémentarités.

(b) Le conseil d'écosystème (*« Ecosystem Council »*) a pour fonction d'améliorer la visibilité du consortium et de s'assurer que les activités (*« activities »*) sont en adéquation avec les tendances du marché. Il propose différentes recommandations afin que le consortium

soit maintenu informé des évolutions de l'environnement. L'Ecosystem Council donne aussi des recommandations sur la politique de communication du consortium.

(c) Le conseil opérationnel (*« Operation Council »*) a pour objectif de proposer des orientations en matière de gestion et juridique. Au moment de nos investigations ce conseil n'était pas fonctionnel mais son existence était planifiée.

Pour finir, il convient de noter qu'une partie des acteurs d'OW2 sont employés par le Consortium OW2. D'autres organisations de l'open source emploient aussi directement leurs membres comme Mozilla ce qui remet largement en question la définition de contributeurs[100] proposée par O'Mahony (2003).

2.5.4. Etude de cas de membres choisis : Experlog et Ubikis.

Etant donné la taille du consortium OW2 et le nombre de participants (près de 70 organisations), il n'était pas envisageable d'étudier et de présenter la stratégie de chaque organisation impliquée dans OW2. Nous avons déjà réalisé une présentation des enjeux de la participation par groupes d'acteurs, dans cette partie nous présenterons les enjeux de la participation à OW2 de deux petites PME (Experlog et Ubikis). Une présentation du rôle de Thales dans le consortium OW2 sera réalisée dans la partie dédiée à l'analyse de la stratégie open source de Thales[101].

A. Les enjeux de la participation d'Experlog.

Pour Experlog, la participation à OW2 est *« important[e] sans être forcément vital[e] (quoique pas négligeable en termes de réseau de relations, et même de chiffre d'affaires généré), et très intéressant[102]. »* Le fondateur explique qu'il est en outre *« très intéressé par l'open-source, pas seulement pour des raisons professionnelles - disons qu'il y a aussi une part de convictions dans la démarche. »* Il ajoute que *« je suis assez proche des idées de Lessig voire de Moglen sur l'évolution de la propriété intellectuelle, et en accord avec une bonne partie de ce que promeut la FSF (quand elle ne force pas trop le trait). J'ai souvent pris des positions idéologiques plus proches du libre que de l'open-source industriel même si*

[100] O'Mahony souligne que « les contributeurs peuvent être sponsorisés par des firmes, mais ils ne sont pas employés par le projet et les relations liées au projet ne sont pas guidées par des relations d'emploi. » (O'Mahony, 2003 : 1179-1180).
[101] A propos de la stratégie de Thales se référer au chapitre 2 de la partie IV.
[102] L'ensemble des informations citées sur Experlog sont issues de Pierre-Yves Gibello, Fondateur, Experlog, entretien en ligne avec l'auteur 31 juillet 2009.

les deux se rejoignent dans une large mesure... Quelque part, c'est un peu comme un engagement politique, au fond ! » Il y a donc une dualité des engagements pour ce participant.

Malgré sa petite taille Experlog participe activement à OW2. La firme était membre du conseil d'administration jusqu'à l'an dernier mais demeure membre du *« technical council »* et est à la tête d'un projet OW2.

Du point de vue technologique, la participation d'Experlog à OW2 représente *« de la veille, de la prospective, et des outils. »* Du point de vue économique, cela représente *« aussi un réseau »* qui a permis à Experlog *« de travailler (en sous-traitance, ainsi que sur projet à financement public). »*

Experlog est membre d'OW2 depuis pratiquement le début (2000), la PME a assisté à la fondation d'ObjectWeb. Le consortium était à l'époque *« assez artisanal (et très local sur Grenoble). »* Experlog a été fondée quasiment en même temps que le consortium OW2, il est par conséquent difficile de mesurer l'impact de la participation d'Experlog à OW2. Mais il est certain que *« OW2 a ouvert pas mal d'opportunités à ExperLog (« networking »), et m'a aidé [à] rester à jour sur le plan technique. »*

B. Les enjeux de la participation d'Ubikis.

Ubikis est une petite PME composée de 5 personnes. Le modèle économique d'Ubikis repose sur *« la promotion d'un projet destiné au développement d'applications nomades. Nous essayons de créer des synergies avec des éditeurs / intégrateurs pour partager nos métiers et faire des applications nomades pour un certain domaine. Après nous partageons les bénéfices*[103]*. »*

Ubikis est entré en 2005 dans OW2 car la firme a mis ses développements en Open source. L'enjeu de la participation d'Ubikis a deux aspects : d'un coté l'entreprise bénéficie de l'image véhiculée par le consortium OW2 et de l'autre c'est une volonté d'*« adéquation avec nos clients (entreprise). »*

Ubikis a une participation très faible à OW2. Sa participation se limite pratiquement au maintien d'Open Mobile IS. Selon, un représentant d'Ubikis, *« Le meilleur retour c'est les contacts avec les membres. »* En ce qui concerne les liens de la firme avec les autres membres : *« [Ils sont] très faible. On a bien essayé au début mais les synergies étaient très faibles. Nous sommes aussi un projet atypique dans OW2 puisque nous sommes en périphérie*

[103] L'ensemble des informations citées sur Ubikis sont issues de Philippe Delrieu, Directeur Technique, Ubikis, entretien en ligne avec l'auteur 13 aout 2009.

du SI [Système d'information]. Nos développements concernent les applis se trouvant en dehors de l'entreprise. Mais je pense que nous n'avons pas fait beaucoup d'effort car on était dans notre dynamique. Je pense que nous serons plus présents dans OW2 avec le développement de la communauté autour d'Open Mobile IS. »

2.5.5. OW2 Consortium : l'enjeu d'une indépendance technologique.

D'un point de vue plus général, OW2 est une tentative de prise d'indépendance technologique d'acteurs de l'industrie du middleware jusqu'ici dominé par des éditeurs américains[104]. Comme le souligne un représentant de Thales au sein d'OW2, *« On a très peu d'éditeur en Europe à part SAP. Effectivement, la plupart des éditeurs de logiciels sont américains. D'où l'intérêt de l'Europe pour les logiciels libres, on ne peut pas concurrencer les américains sur le marché du logiciel, c'est pour ça que l'Europe se tourne vers le logiciel libre. Ca conjugué à la montée de l'économie orientée services. Clairement le positionnement de l'Europe, c'est parce que l'on n'est pas capable de se positionner face à l'industrie américaine sur le marché du logiciel propriétaire[105]. »*

En outre, la participation d'organisations chinoises dans OW2 n'est pas surprenante. *« La Chine est « anti-américains » et foncièrement européen. [En Chine], la notion de propriété intellectuelle est très différente. Lorsque l'on parle d'open source avec un universitaire, un membre du ministère chinois, un industriel, etc. ils associent toujours open source et business model[106]. »*

OW2 est donc le résultat de la convergence de deux stratégies visant à instaurer une indépendance technologique au niveau d'une partie *clé* du système d'information : le middleware. L'Europe et la Chine ont des intérêts communs. Contrairement aux Etats-Unis, l'Europe et la Chine ont plutôt des intégrateurs et peu d'éditeurs.

OW2 est un des éléments visant à renforcer la concurrence des entreprises européennes sur le marché mondial de l'informatique et des services. Pour certains grands acteurs de l'industrie, il n'était pas possible d'être dépendants technologiquement d'autres pays pour des composants aussi importants que le middleware. L'une des principales missions d'OW2 est de créer une infrastructure middleware libre. D'après le représentant de Thales au sein du board

[104] Fuggetta défend l'idée qu'en Europe l'open source est vu comme une opportunité car il permet d'une part, de contrecarrer la domination technologique américaine et d'autre part, d'avoir une meilleure autonomie technologique pour les Etats (Fuggetta 2003: 78).
[105] Julie Marguerite, Open Source Architect, Thales D3S, entretien en face à face avec l'auteur, 6 juin 2007.
[106] Grégory Lopez, Responsable du Centre de Compétences Open Source, Thales D3S, entretien avec l'auteur, 23 janvier 2007.

d'OW2, *« le Middleware est un objet technique qui fait l'intermédiaire entre le matériel et les couches hautes. Il est indispensable mais ce n'est pas là où on créé de la valeur. Thales, Bull, etc. ne sont pas des éditeurs de logiciels. Dans l'environnement propriétaire il y avait une dépendance américaine. Cette dépendance est technologique, ce n'est pas pour faire du « cocorico »… Les composants middleware ont une faible valeur ajoutée mais ils sont essentiels. Dans OW2, on partage le risque de mise à jour du middleware*[107]*. »*

OW2 Consortium est une des quatre plus influentes organisations de l'industrie open source avec Eclipse, Apache et la Linux Foundation. L'objet du consortium OW2 n'est pas intrinsèquement orienté vers la création de logiciels proposant de nouveaux usages. Le consortium OW2 vise plutôt à proposer des composants que les participants combineront avec d'autres technologies libres et non libres pour créer des solutions sur-mesure ou des systèmes d'informations. Selon Serge Druais de Thales D3S, *« Pour la communauté OW2, la focalisation est en effet le middleware et comme son nom l'indique, on est loin de l'utilisateur final et donc de l'innovation dans les usages*[108]*. »*

Il convient de noter qu'historiquement l'open source a émergé dans le domaine de l'infrastructure logicielle (Sharma et al. 2002: 21). D'autre part, il est clair que les autres consortiums open source ne sont pas orientés vers la création de nouveaux usages. Dans Eclipse, une communauté d'entreprises ayant pour but de fournir des plates-formes et des applications ouvertes pour la production de logiciels, nous sommes également loin des couches hautes même si *« Eclipse monte dans la chaîne de valeur*[109]*. »* Le noyau Linux est de même très loin des couches hautes. L'Apache Foundation est de même loin de l'utilisateur final et s'adresse beaucoup plus aux professionnels : utiliser un serveur Apache nécessite des compétences particulières n'étant pas à la portée de n'importe quel utilisateur.

Ces coopérations inter-firmes portent sur des couches intermédiaires de l'infrastructure logicielle et peu sur des couches hautes. En définitif, dans l'ensemble des organisations ayant une orientation industrielle, nous sommes dans une logique de composants ayant peu de valeur ajoutée, c'est toutefois grâce à ces composants que la valeur est créée par les entreprises et comme le souligne un spécialiste de l'open source : *« l'open source n'est pas métier, il est technique*[110]*. »*

[107] Serge Druais, Directeur de la Recherche et de l'Innovation, Thales D3S, entretien téléphonique avec l'auteur, 5 aout 2009.
[108] Ibid.
[109] Grégory Lopez, Responsable du Centre de Compétences Open Source, Thales D3S, entretien en face à face avec l'auteur, 23 janvier 2007.
[110] Ibid.

Chapitre 2. LA MODELISATION DES REGIMES DE L'OPEN SOURCE.

Dans les cas étudiés, les paramètres organisationnels sélectionnés sont présents mais à des niveaux d'importance différents. A partir de ces cas et des caractéristiques dominantes de ces derniers, nous avons bâti une typologie *des régimes de l'open source*. En d'autres termes, nous avons construit une typologie des formes d'action collective impliquées dans la production de logiciels libres. Nous restituerons le cheminement logique conduisant à la construction de ces régimes. Nous définirons d'abord les paramètres organisationnels sur lesquels repose notre analyse [1.]. Puis à partir d'une approche basée sur de multiples cas nous génèrerons des idéaux-types [2.]. Ensuite, nous détaillerons le modèle du pentagone [3.]. Et enfin, nous mobiliserons ce modèle pour décrire les formes hybrides de l'open source [4.].

1. La définition de paramètres organisationnels.

Dans cette section, nous détaillerons les paramètres organisationnels sélectionnés afin de construire notre typologie des régimes de l'open source. Nous avons recherché les éléments permettant de distinguer les différentes formes organisationnelles sélectionnées car la littérature fait souvent usage du mot *« community »* pour décrire une classe de phénomènes différente. Les paramètres dominants dans les cas sont résumés dans le tableau ci-dessous.

Tableau 18 : Synthèse des paramètres dominants dans les cas.

	Acteurs principaux*	Mode de Rétribution*	Mode de financement principal*	Nature de l'organisation*
Mandriva	Entreprise	Rémunération Directe	Prestation de Services	Formelle
Kexi	Utilisateurs-développeurs	Bénévolat	Propres Ressources	Informelle
Freeworks	Utilisateurs-développeurs	Bénévolat	Propres Ressources	Formelle
OpenOffice.org	Entreprise	Rémunération Directe	Propres Ressources	Informelle
OW2 Consortium	Entreprise	Rémunération Directe	Propres Ressources	Formelle

*Caractère dominant de l'organisation.

Le mode de financement est le premier paramètre permettant de distinguer nos cas. Il est étroitement lié avec le but des organisations de l'open source. Quatre des cinq cas ont un financement essentiellement basé sur les propres ressources des participants comme cela a déjà été démontré par von Hippel et von Krogh (2003). Mandriva est la seule organisation dont le financement est basé sur de la prestation de services et la vente de produits. De

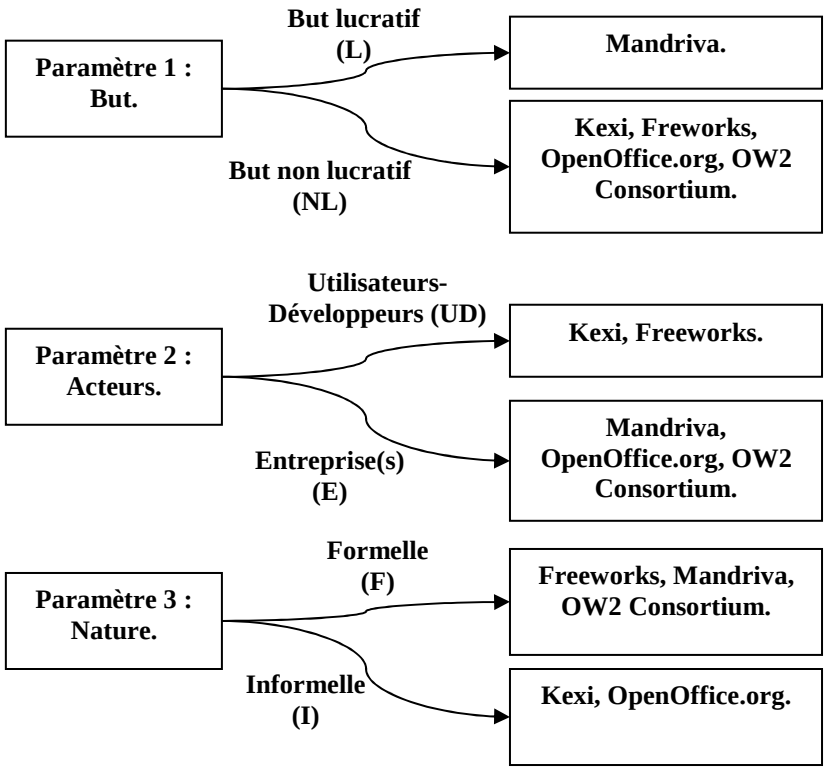
manière plus globale, ce premier paramètre distingue les organisations à but lucratif (entreprises) et les organisations à but non lucratif identifiées dans la littérature sous le concept de *« community »*.

Le type d'acteurs est le second paramètre organisationnel. Kexi et Freeworks mettent en relation principalement des utilisateurs-développeurs. Les autres cas ont principalement pour membre des employés d'une (Mandriva) ou plusieurs entreprises (OpenOffice.org et OW2 Consortium). Ce second paramètre sépare nos cas en deux catégories. Il convient de souligner qu'ici le mode de contribution et rétribution des acteurs est étroitement lié avec le type d'acteurs. Ainsi, les membres employés perçoivent une rémunération directe tandis que les utilisateurs-développeurs basent leur financement sur du bénévolat. Par conséquent, le type d'acteurs et le mode de contribution et rétribution peuvent être confondus en un seul et même paramètre

La nature de l'organisation est le troisième paramètre. Nos cas sont séparables en deux catégories : les organisations formelles (c'est-à-dire avec une existence juridique) et les organisations informelles (c'est-à-dire sans existence au niveau légal).

Pour résumer ces trois paramètres et leur impact sur le partitionnement des cas nous suggérons les représentations graphiques ci-dessous.

Figure 20 : Schéma de l'application séparée des paramètres organisationnels.



2. De l'étude de cas multiple à la génération d'idéaux-types.

Nous proposons une mise en évidence des paramètres organisationnels dominants de chacun des cas.

Tableau 19 : Paramètres organisationnels dominants dans les cas.

	Paramètre 1 : But		Paramètre 2 : Acteurs		Paramètre 3 : Existence	
	Lucratif	Non lucratif	Utilisateurs-Développeurs	Entreprise(s)	Formelle	Informelle
Mandriva	X			X	X	
Kexi		X	X			X
Freeworks		X	X		X	
OpenOffice.org		X		X		X
OW2 Consortium		X		X	X	

En reprenant, les paramètres organisationnels définis et en exprimant chacun de nos cas en fonction de ceux-ci, nous sommes en mesure de dériver pour chaque cas étudié une forme simplifiée[111] ou un idéal type que nous nommerons de manière à faire la distinction entre les différentes formes d'organisations de l'open source.

Mandriva	→	**(L, E, F)**	**Firme de l'open source**
Kexi	→	**(NL, UD, I)**	**Communauté d'utilisateurs-développeurs**
Freeworks	→	**(NL, UD, F)**	**Association d'utilisateurs-développeurs**
OpenOffice.org	→	**(NL, E, I)**	**Communauté inter-organisations**
OW2 Consortium	→	**(NL, E, I)**	**Consortium de l'open source**

Cinq formes d'organisations peuvent être distinguées dans l'open source : (1) la firme de l'open source, (2) la communauté d'utilisateurs-développeurs, (3) l'association d'utilisateurs-développeurs, (4) la communauté inter-organisations et (5) le consortium de l'open source. Dans la prochaine section, nous définirons les propriétés de chaque régime.

3. La caractérisation des régimes de l'open source.

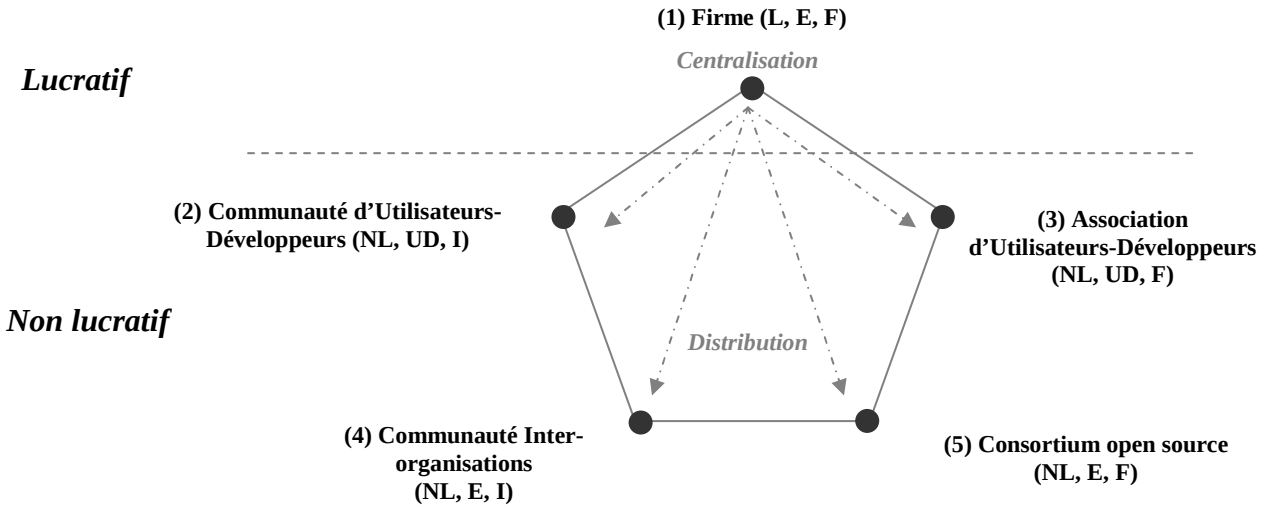
Afin de synthétiser notre typologie des régimes de l'open source et de la rendre plus facilement mémorisable, nous adopterons ce que nous nommons *le modèle du pentagone*[112]

[111] Où : L = Lucratif ; NL = Non Lucratif ; UD = Utilisateurs-Développeurs ; E = Entreprises ; F = Formelle ; I = Informelle.

[112] La première version du modèle du pentagone fut présentée lors de la conférence Euram 2007 (Benkeltoum 2007). Cette version intègre les remarques des relecteurs anonymes et des participants que nous remercions.

(représenté dans le graphique ci-dessous[113]). Ensuite nous expliquerons ce que chaque idéal type représente.

Figure 21 : Le modèle du pentagone.



3.1. La firme de l'open source.

Les firmes (L, E, F) ont différents niveaux d'implication dans l'open source comme cela a déjà été démontré (Grand et al. 2004). Le but de ce paragraphe n'est pas de recenser les différents rôles que peuvent avoir les firmes mais plutôt de décrire le profil général de la firme de l'open source synthétisé à partir de nos observations empiriques. Nous définissons la firme de l'open source comme une entité économique composée de personnes et de biens ayant bâti ou adapté tout ou partie de son modèle économique à l'open source et étant organisée pour fournir des biens et/ou des services pour l'industrie informatique. Il existe une grande variété de modèles économiques adaptés aux logiciels libres[114]. Le développement est de type centralisé et réalisé par des individus liés par une solidarité hiérarchique

3.2. La communauté d'utilisateurs-développeurs.

La communauté d'utilisateurs-développeurs constitue ce que nous considérons être le modèle organisationnel de base à l'origine des régimes de l'open source. Il s'agit d'une organisation composée d'acteurs distribués contribuant en leur propre nom et liés par une

[113] Notre travail s'est largement inspiré de l'ouvrage de H. Mintzberg (1982) en termes de logique de raisonnement et de présentation.
[114] Nous reviendrons plus en détail sur les modèles économiques des firmes de l'open source dans la partie IV.

solidarité d'usage, d'apprentissage et de conception. Les acteurs ne sont pas rémunérés pour le travail réalisé. Ce type d'organisation n'a pas existence du point de vue légal cela signifie qu'elle ne peut pas (dans la plupart des cas) accepter les dons des utilisateurs surtout lorsqu'il y a plusieurs individus car un problème de répartition apparaît. L'objectif des communautés d'utilisateurs-développeurs est de développer et promouvoir un ou plusieurs logiciels libres.

3.3. L'association d'utilisateurs-développeurs.

L'association d'utilisateurs-développeurs a des similarités avec la communauté d'utilisateurs-développeurs dans la mesure où les individus sont principalement liés par une solidarité d'usage, d'apprentissage et de conception. En revanche, l'association d'utilisateurs-développeurs a deux particularités. Premièrement, elle dispose d'une existence légale. Les structures juridiques variant beaucoup d'un pays à l'autre. Par exemple : une association de loi 1901 en France, une *« non-profit corporation »* aux Etats-Unis, une « *Eingetragener Verein (e.V.) »* en Allemagne, etc. Cette existence légale confère à cette organisation la possibilité d'accepter les dons des utilisateurs mais aussi d'avoir un ou plusieurs permanents rémunérés par l'organisation. Deuxièmement, dans la plupart des cas les associations d'utilisateurs-développeurs regroupent des individus qui se sont déjà rencontrés[115]. Les associations d'utilisateurs-développeurs sont donc plus *privées* que les communautés d'utilisateurs-développeurs.

3.4. La communauté inter-organisations.

La communauté inter-organisations est composée d'organisations dont la plupart sont des entreprises. Ces organisations sont représentées par des développeurs employés n'étant pas forcément utilisateurs du produit. Les organisations sont liées par une solidarité économique et/ou technologique. Du point de vue économique, elles sont motivées par une mutualisation des coûts et/ou la lutte contre un monopole. Du point de vue technologique, elles maintiennent une base de code *« générique[116] »* ou un logiciel orienté utilisateur final de type *« commodité[117] »*. Il s'agit d'organisations informelles : c'est à dire sans existence du point de vue légal. Cette situation pose des problèmes notamment en ce qui concerne le financement

[115] Il y a un véritable phénomène de cooptation dans ces organisations.
[116] La notion de composant générique fait référence à une brique technologique standardisée dont la technologie est connue et maîtrisée.
[117] Une commodité est un logiciel orienté utilisateur final satisfaisant un usage connu et standardisé.

des activités de la communauté puisque celle-ci repose uniquement sur les ressources des organisations participantes.

Organisation informelle ne signifie pas pour autant que les communautés inter-organisations ne sont régies par aucune règle. Dans ces organisations, les questions de droits d'auteurs sont encadrées de manière très précise grâce à des contrats comme par exemple un contrat de partage de copyright (cf. le cas OpenOffice.org). Les acteurs impliqués dans la communauté inter-organisations y participent car ils agissent directement ou indirectement sur leur développement économique. Ces entreprises sont notamment motivées par la réouverture de certains marchés où la domination d'un ou de quelques éditeurs empêchent la concurrence. Une communauté inter-organisations peut être définie comme une organisation informelle créée ou passée sous le contrôle d'une ou plusieurs organisations dans le but de diffuser un logiciel libre ou un standard ouvert dans l'industrie informatique.

3.5. Le consortium de l'open source.

Le consortium de l'open source se compose d'organisations (principalement des entreprises) dans ce sens elle est assez proche de la précédente forme organisationnelle. L'aspect le plus différenciant est le fait que le consortium dispose d'une existence juridique. Cela a de deux impacts importants. D'un côté, le consortium détient des ressources propres conférant au consortium une liberté d'action plus grande que la communauté inter-organisations. D'un autre côté, le consortium peut se doter d'un bureau de gestion ou de permanents chargés de promouvoir les intérêts du consortium ce que la communauté inter-organisations ne peut pas faire.

Le développement d'affaires fait clairement parti des objectifs du consortium. En tout cas, dans le cas OW2 Consortium. Toutefois, le consortium ne réalise pas de vente ni de prestations de services cela signifie que le financement de ces consortiums repose sur la contribution des organisations participantes.

Les organisations impliquées recherchent un meilleur contrôle de leur environnement technologique et par conséquent une meilleure autonomie vis-à-vis des éditeurs de logiciels. Bien souvent les consortiums développent des briques technologiques génériques avec également la volonté de réouvrir la concurrence sur certains marchés peu concurrentiels voire monopolistiques. La participation au consortium est souvent conditionnée à la signature d'un contrat de membre prévoyant le versement de droits proportionnels à la taille de l'entreprise et

au niveau de participation qu'elle souhaite avoir dans le consortium. Dans la plupart des cas, le contenu du contrat est dédié à la gestion des droits d'auteurs.

4. Des idéaux types au service de la description des formes hybrides de l'open source.

Notre typologie doit davantage être vue comme un aide à la description des formes organisationnelles de l'open source que comme cinq configurations incompatibles entre elles. Bien au contraire, nous considérons qu'elle doit être considérée comme des *« types purs »* pour reprendre la terminologie et la logique de H. Mintzberg. Nous pensons donc que ces configurations peuvent *« être considérées comme la base permettant de décrire les structures hybrides »* de l'open source (Mintzberg 1982: 412-13). Notre typologie est une vision idéalisée de l'open source car bon nombre d'organisations sont à mis chemin entre une ou plusieurs formes décrites. Dans les deux prochaines sections, nous proposons d'abord de relire nos cas sous l'angle de l'hybridation [4.1.] et ensuite nous utiliserons le modèle du pentagone pour décrire d'autres organisations de l'open source [4.2.].

4.1. Relecture des cas sous l'angle de l'hybridation.

Mandriva dispose du système de fonctionnement d'une firme (L, E, F) pour sa majeure partie. En revanche, Mandriva gère aussi de manière informelle une communauté de Cookers (NL, UD, I) qui contribuent à la distribution Mandriva Linux.

Kexi est principalement rattachable à la communauté d'utilisateurs-développeurs (NL, UD, I) car la majorité des participants sont des utilisateurs-développeurs bénévoles. Toutefois, un développeur est rémunéré par une firme notamment pour le temps passé sur le développement de Kexi. Il s'agit de ce que O'Mahony nomme *« sponsored contributor »* (O'Mahony 2002: 190; 2003: 1180; 2005: 403). Kexi a donc un fonctionnement très proche de celui d'une communauté d'utilisateurs-développeurs avec quelques petites caractéristiques de la firme. Nous serions tentés de qualifier Kexi de communauté d'utilisateurs-développeurs mono-sponsor en nous basant sur la notion de *« sponsor »* proposée par West et O'Mahony (2005, 2007).

Freeworks est une association ayant également un fonctionnement particulier puisque dans une large mesure elle fonctionne comme une association d'utilisateurs-développeurs (NL, UD, F) et elle dispose aussi de certaines caractéristiques particulières de la firme (L, E,

F). Les membres de Freeworks réalisent de la prestation de services sous le nom de l'association. Nous voyons bien à travers les cas Kexi et Freeworks que le développement de logiciels libres a un coût (Fuggetta 2003: 86) et que le financement est une réelle question de fond. Nous traiterons la question du financement dans la prochaine partie de notre travail car cela nécessite une analyse approfondie.

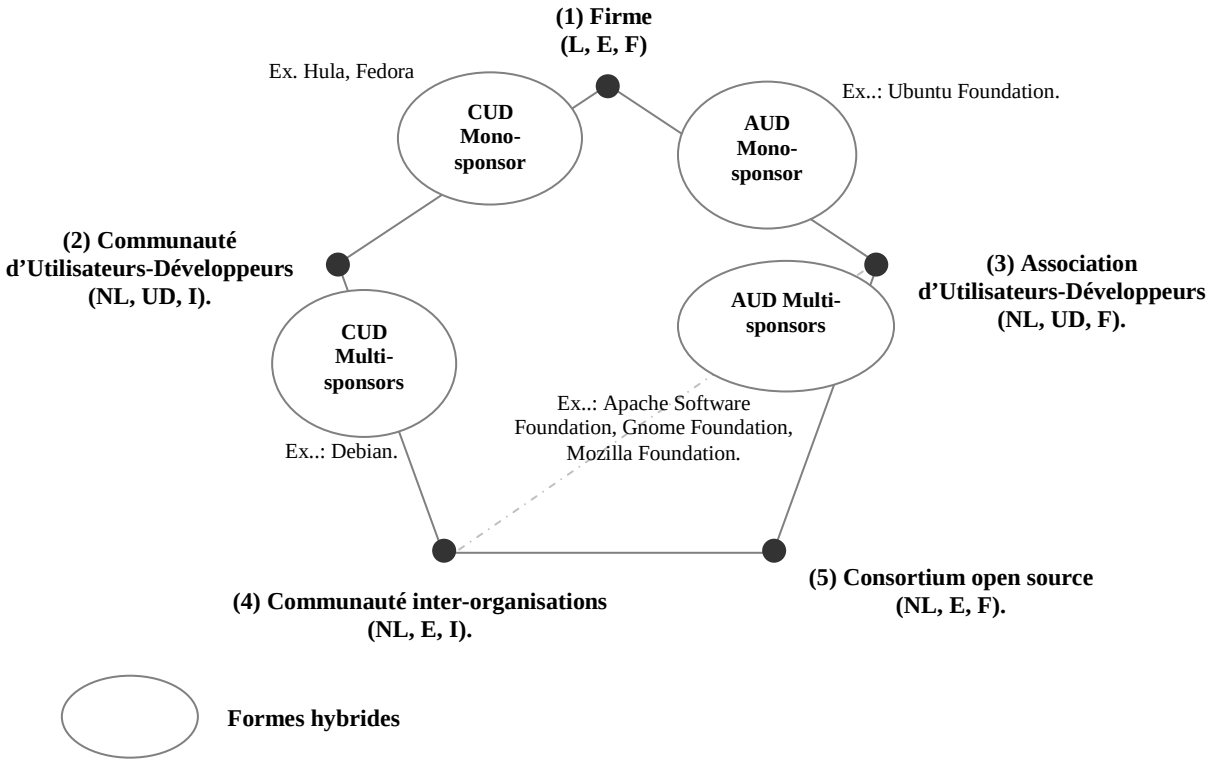
OpenOffice.org est une communauté principalement composée d'employés de grandes entreprises de l'industrie informatique. Mais elle comprend aussi une partie de contributeurs indépendants ayant un rôle important en particulier sur les projets linguistiques : OpenOffice.org comprend près de 90 projets linguistiques.

OW2 est un consortium fondé par de grands acteurs de l'industrie informatique. OW2 comprend également une partie de contributeurs indépendants pour la plupart consultants. La gouvernance d'OW2 est toutefois clairement réservée aux entreprises en mesure de fournir des effectifs permanents et des fonds au consortium.

4.2. L'utilisation du modèle du pentagone pour décrire les formes hybrides.

Le modèle du pentagone permet de décrire les formes d'organisations existantes de l'open source. En reprenant des exemples d'organisations de l'open source étudiées de manière moins profonde, nous détaillerons quelques formes hybrides.

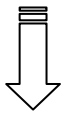
Figure 22 : Le modèle du pentagone et la description de formes hybrides



Par exemple, l'Apache Software Foundation (ASF) est considérée comme une communauté sponsorisée par O'Mahony (2002). En appliquant notre typologie nous disons que l'ASF est à mis chemin entre l'association d'utilisateurs-développeurs (NL, UD, F) et la communauté inter-organisations (NL, E, I). D'après l'ASF, aucun membre de l'organisation n'est rémunéré par la fondation pour le travail qu'il réalise. En revanche, il y a beaucoup de *« committers*[118] *»* rémunérés par des entreprises ou des institutions souhaitant améliorer ou soutenir Apache[119].

Question de Recherche 2 :

QR 2 : Quelles sont les expansions de ce modèle, sur quels critères distinguer les formes résultantes et comment suivre les transformations organisationnelles dans l'open source ?



Thèse 2 :

TH 2 : **La communauté d'utilisateurs-développeurs est le modèle racine à l'origine de différents régimes de production celui-ci a connu différentes expansions (synthétisées dans le modèle du pentagone) dues à l'implication croissante des firmes dans le phénomène open source.**

[118] L'expression *« committer »* signifie littéralement le fait de déposer du code dans un système de gestion de code source en ligne. Toutefois par extension, un *« committer »* désigne un programmeur contribuant au développement d'un logiciel libre et ayant les droits d'écriture pour déposer ou modifier du code dans ce système.

[119] Pour de plus amples informations sur le cas Apache se référer à (Fielding 1999).

Dans la troisième partie de notre thèse [Partie III], nous avons en premier lieu vu que la littérature signalait le manque de critères afin de distinguer les organisations de l'open source (O'Mahony 2007: 144). Puis, nous avons défendu l'idée que les notions de *« community »* et de *« projet »* étaient trop vagues pour pouvoir distinguer les organisations impliquées dans le développement de logiciels libres. Cette variété fut aussi bien attestée par les praticiens au cours de nos entretiens.

D'autre part, en utilisant la technique de l'échantillonnage théorique (Eisenhardt et Graebner 2007: 27), nous avons sélectionné cinq cas (Kexi, Freeworks, Mandriva, OpenOffice.org et OW2) suffisamment hétérogènes pour rendre compte de la diversité des régimes de production rencontrés. Cette sélection s'est basée sur quatre critères : le type d'acteurs, le mode de financement, le système de rémunération et le caractère formel ou informel de l'organisation. Ces critères ont été choisis en nous basant sur les conclusions des parties précédentes [Thèse 1], nos observations empiriques et la littérature correspondante.

Les cinq études de cas sélectionnées, nous ont permis de dériver une modélisation des régimes de l'open source nommée modèle du pentagone [Chapitre 2]. Ce modèle contient cinq idéaux-types d'organisations grâce auxquels nous caractérisons les organisations existantes qui sont généralement hybrides : c'est-à-dire entre un ou plusieurs idéal-type. De plus, ce modèle peut être employé pour suivre les transformations organisationnelles d'un point historique.

Au final, nous montrons que la communauté d'utilisateurs-développeurs est un modèle souche qui a connu différentes expansions : (la firme, l'association d'utilisateurs-développeurs, la communauté inter-organisations et le consortium). Expansions dues à l'implication croissante des firmes dans le phénomène open source [Thèse 2].

PARTIE IV. LES STRATEGIES DANS L'OPEN SOURCE : DES COMBINAISONS NOUVELLES ENTRE LE MARCHAND ET LE NON-MARCHAND.

Avant d'analyser la dynamique des régimes de l'open source : c'est-à-dire l'examen des transformations stratégiques[120], économiques et technologiques des organisations de l'open source [Chapitre 3] nous avons besoin de écrire deux éléments. D'une part, il convient au préalable d'étudier et explorer les stratégies économiques des acteurs de l'open source [Chapitre 1]. D'autre part, il faudra également détailler les stratégies industrielles et coopératives dans l'open source [Chapitre 2].

Chapitre 1. VALORISATION ECONOMIQUE DE L'OPEN SOURCE ET STRATEGIES.

Dans le présent chapitre, nous allons tout d'abord nous intéresser aux différents modèles de valorisation des logiciels libres permettant aux entreprises mais aussi aux différentes organisations (régimes de l'open source) de financer le développement des logiciels libres.

La littérature fait état de deux catégories de modèles de valorisation économique pour les logiciels libres. Il y a d'un côté les modèles économiques *ouverts* : cela signifie que la base logicielle employée est libre et que le logiciel produit reste libre ; d'un autre côté, il y a des modèles économiques hybrides qui conjuguent technologies libres et fermées [1.].

A travers nos investigations empiriques, nous avons identifié une troisième catégorie de modèles économiques basés sur différentes stratégies de verrouillage. En d'autres termes : la base logicielle utilisée est libre et le logiciel produit est fermé ou quasi-fermé [2.].

1. Les stratégies de valorisation ouvertes et hybrides.

1.1. Les stratégies ouvertes.

Dahlander répertorie cinq manières d'exploiter une innovation provenant des logiciels libres : (1) le consulting : la firme propose des services de conseil liés à un logiciel libre ; (2)

[120] Nos développements sur les stratégies dans l'open source s'inscrivent dans la lignée du travail initié par F. Aggeri invitant à faire dialoguer les visions « *stratégique* » et « *ingénierique* » de la gestion (Aggeri 2008: 7).

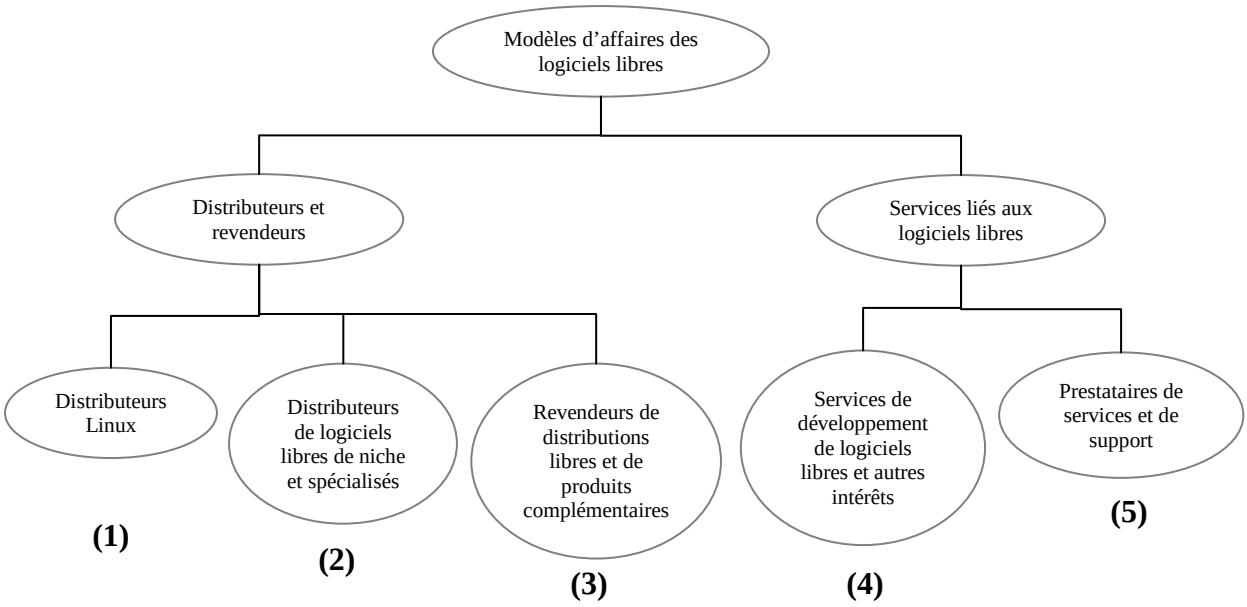
la formation : la firme réalise des formations à un logiciel libre ; (3) le support : la firme offre des services de support client pour l'utilisation d'un logiciel libre ; (4) le licensing : la firme commercialise une licence d'utilisation pour un logiciel libre due à une valeur ajoutée apportée comme des applications non libres ; (5) la boîte noire : la firme embarque du logiciel open source dans un matériel (Dahlander 2004: 6).

Pal et Madanmohan proposent une typologie des différents modes de valorisation qu'une firme peut avoir. (1) Un seul logiciel libre et un seul marché : c'est une stratégie simple et plutôt adaptée aux petites entreprises. (2) Un seul logiciel libre et de multiples marchés : cette stratégie consiste à avoir une seule technologie open source mais applicable sur différents marchés. La technologie open source sert de socle pour bâtir des solutions logicielles complétées par des composants non libres. Il s'agit d'une stratégie plutôt adaptée aux entreprises ayant de nombreux produits mais une seule technologie de base. (3) De multiples logiciel libre et un seul marché : il s'agit d'une stratégie consistant à lancer et gérer de multiples initiatives open source. Cette stratégie correspond davantage aux entreprises leaders sur le marché d'un produit ou d'une technologie particulière. (4) De multiples logiciels libres et de multiples marchés : cette stratégie correspond aux grandes entreprises ayant les capacités de gérer de multiples logiciels libres et de s'attaquer à de multiples marchés.

Muselli suggère le modèle de la *« valorisation indirecte »*, un modèle basé sur de la prestation de services (Muselli 2004a: 7). De même, Onetti et Capobianco (2005) décrivent le modèle *« GPL/LGPL »* : un modèle d'affaires basé essentiellement sur de la prestation de services comme (la maintenance, les développements spécifiques, le conseil et la formation) (Onetti et Capobianco 2005: 224).

Spiller et Wichmann proposent un schéma résumant les principaux modèles d'affaires des entreprises impliquées dans l'open source (Spiller et Wichmann 2002: 42).

Figure 23 : Les modèles d'affaires de l'open source selon Spiller et Wichmann (2002).



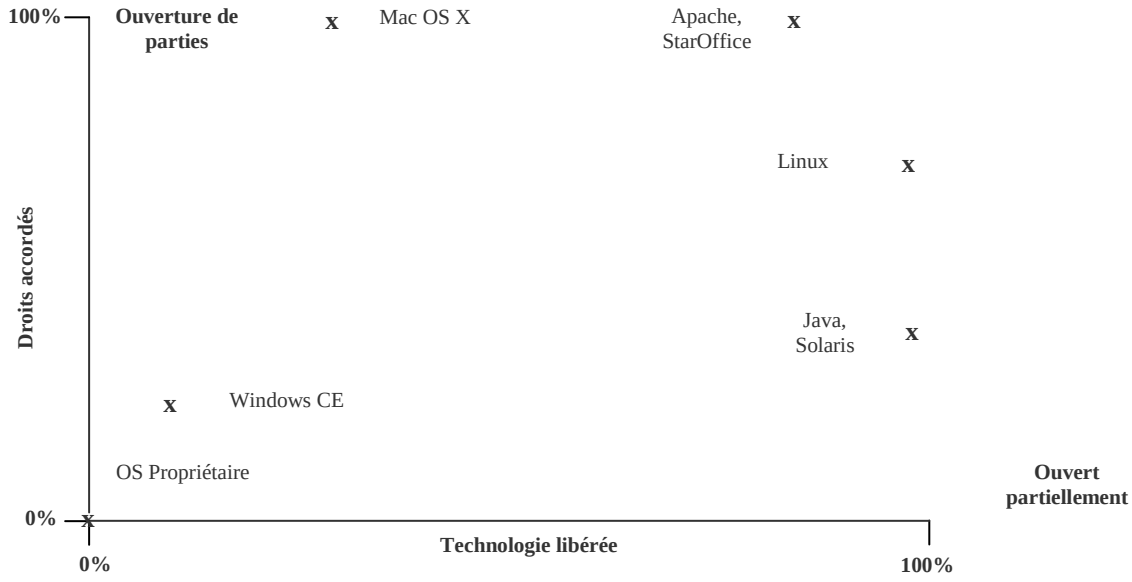
Nous résumerons chaque modèle d'affaire suggéré par les auteurs. (1) Les distributeurs Linux : l'objectif de ces entreprises est d'obtenir les dernières versions du noyau ainsi que d'autres logiciels libres afin de proposer un tout cohérent. Cet élément est obtenu grâce au travail d'adaptation réalisé. D'après Spiller et Wichmann, ces entreprises ont trois missions (a) collecter les fichiers du noyau linux et d'autres logiciels libres ; (b) tester, adapter et optimiser les logiciels de manière qu'ils soient le plus performants et stables ; (c) créer des outils facilitant l'installation et l'utilisation. (2) Les distributeurs de logiciels libres de niche et spécialisés : cette catégorie d'entreprises développe des logiciels libres (hors systèmes d'exploitation). (3) Les revendeurs de distributions libres et de produits complémentaires : ces entreprises vendent les produits logiciels des distributeurs ainsi que de la documentation et des informations sur des logiciels libres. (4) Les prestataires de services de développement de logiciels libres et autres intérêts : cette catégorie d'entreprises comprend deux sous types (4a) les places de marché ayant pour objectif de faire rencontrer des organisations ou des individus souhaitant obtenir des solutions adaptées à leurs besoins et d'autre part des développeurs open source. (4b) Les organisateurs de conférences ayant pour objectif de promouvoir les opportunités commerciales de l'open source. (5) Les prestataires de services et de support : cette catégorie d'entreprises comprend des prestataires de service (exemples : consulting, maintenance, formation, support client etc.) (Spiller et Wichmann 2002: 42-48).

1.2. Les stratégies hybrides.

Il est évident que les *mondes* du logiciel libre et du logiciel fermé ne sont pas séparés par une frontière opaque et ne sont pas exclusifs. Certains auteurs se sont intéressés aux stratégies d'entreprises qui *à cheval* entre le libre et le fermé.

Ainsi, J. West a décrit deux stratégies à mi-chemin entre le monde fermé et l'open source : (1) *« opening parts »* : consistant à libérer une partie seulement du logiciel et à garder l'autre plus stratégique secrète (source de différenciation) ; (2) *« partly open »* : consistant à *« ouvrir »* une technologie en s'assurant d'avoir des restrictions quant à l'utilisation de celle-ci. West donne plusieurs exemples de sociétés menant ce type de stratégies et propose un graphique permettant de visualiser ces stratégies (West 2003: 1279-80).

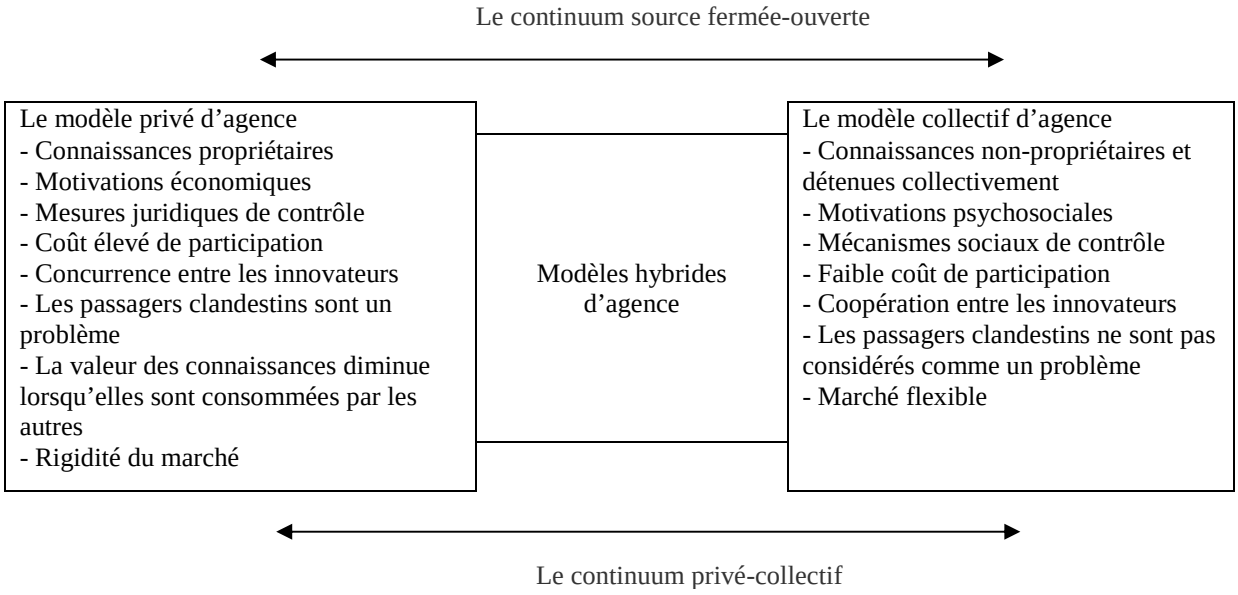
Figure 24 : Droits de l'utilisateur sous les licences open et quasi-open source[121] (West, 2003).



J. P. Ulhoi, quant à lui, a qualifié d'hybrides les stratégies entre le fermé et le libre. Ulhoi propose le schéma reproduit ci-dessous pour caractériser ce qu'il considère être des modèles hybrides (Ulhoi 2004: 1096).

[121] Java désormais devrait être déplacé vers le haut puisqu'il a été libéré sous General Public License (GPL) par Sun Microsystems.

Figure 25 : Stratégies hybrides selon Ulhoi (2004).



J. P. Ulhoi s'est largement inspiré des travaux de von Hippel et von Krogh (2003) sur le *« Private-Collective Model »*.

Bonaccorsi et al. ont étudié les modèles économiques de 146 firmes de l'industrie italienne du logiciel. Ces auteurs trouvent qu'une écrasante majorité (138 sur 146) de firmes bâtissent des modèles économiques hybrides couplant à la fois des moyens classiques de valorisation : (la cession de licences) et des revenus beaucoup plus liés aux logiciels libres (Bonaccorsi et al. 2006: 1085). Cette étude démontre que la majorité des firmes couplent modèles de valorisation libres et non libres. Nous vous proposons de passer en revue les différentes stratégies hybrides décrites dans la littérature.

Pal et Madanmohan ont présenté une stratégie consistant à couper un logiciel en deux parties : (1) une partie open source en tant que plate-forme ; (2) une partie est fermée comme élément servant à l'utilisateur final. Les parties (1) et (2) étant reliés par des interfaces (Pal et Madanmohan 2002: 9-10).

Muselli suggère deux stratégies hybrides. (1) La première est intitulée modèle de *« valorisation directe d'un produit sur deux marchés discriminés »*. Cette stratégie consiste à adopter une double licence pour deux catégories d'utilisateurs (Muselli 2004a: 13). Il n'est valable que pour les composants et non les produits orientés utilisateur final (Muselli 2004a: 15). (2) La seconde est intitulée modèle de *« valorisation directe par vente de produits complémentaires (ou liés) sur un marché unique »*. Il s'agit de proposer deux logiciels avec deux licences différentes pour un même marché. Le logiciel libre a pour but de satisfaire *« un*

besoin basique des utilisateurs du marché » et le second répond à des *« besoins plus évolués de la part des utilisateurs »*. (Muselli 2004a: 18).

Onetti et Capobianco décrivent deux stratégies hybrides en se basant sur différentes catégories de licences libres. Premièrement, il y a (1) le modèle le modèle *« BSD/Apache »* : étant donné que ces licences permettent d'utiliser le code d'un logiciel libre partiellement ou totalement pour en créer un nouveau ou l'adapter à un autre logiciel (libre ou non libre), ces entreprises tirent leurs revenus à la fois de la vente de licences et de services associés. Deuxièmement, il y a (2) le modèle du *« Dual Licensing »* : les entreprises réalisant ce modèle économique perçoivent leurs revenus sur de la vente de licences. Les entreprises clientes utilisant la version non libre ne souhaitent pas que leurs développements soient contaminés par le caractère *« viral*[122] *»* d'une licence. (Onetti et Capobianco 2005: 224).

Rannou et Ronai parlent de stratégies de double licences mais sous un angle différent par rapport à ce qui peut être proposé par les autres auteurs : ils soulignent que des entreprises libèrent une ancienne version d'un logiciel pour faire connaître le produit et vendent la version plus récente sous une licence non libre (Rannou et Ronai 2003a: 109).

1.3. Synthèse des modèles économiques ouverts et hybrides.

D'après la littérature existante et nos investigations empiriques, les modèles d'affaires open source peuvent être classés en trois catégories : (1) les modèles économiques d'éditeurs, (2) d'intégrateurs et (3) de fabricants de matériel.

1.3.1. Les modèles économiques des éditeurs.

Les modèles de valorisation des éditeurs de logiciels libres peuvent être séparés en deux catégories de logiciels : les composants (ou briques technologiques) et les produits utilisateurs finaux.

A. Les produits utilisateurs finaux.

Les produits à destination des utilisateurs finaux (*« end-user products »*) sont valorisés à travers trois modèles : (1) le modèle du produit d'appel, (2) le modèle du produit complémentaire et enfin (3) le modèle de la souscription.

[122] Le caractère *« viral »* désigne le fait que toute modification ou incorporation de code distribué sous certaines licences (GPL principalement) doit également être distribué en adoptant la même licence que l'œuvre initiale.

Le modèle du produit d'appel consiste à diviser le produit en deux produits valorisés à travers le dual licensing ou deux licences différentes (Rannou et Ronai 2003b; Spiller et Wichmann 2002). La version non libre est la dernière version de l'éditeur, elle contient les dernières mises à jour du produit et l'éditeur libérera les anciennes versions du produit de manière à le faire connaître et utiliser. Cette stratégie utilise le temps comme discriminant entre une version compilée complète et une version libre en quelque sorte obsolète. L'entreprise Precision Insight utilise ce modèle économique.

Le modèle du produit complémentaire consistera en la proposition de deux versions toujours selon deux licences différentes (dual licensing) mais cette fois la version libre sera à jour et la version non libre intégrera des éléments additionnels (Muselli 2004a). Il s'agit d'utiliser les besoins comme discriminant entre deux versions : la version libre vise à satisfaire des besoins basiques non élaborés tandis que la version non libre vise à satisfaire des besoins plus sophistiqués. Plusieurs firmes utilisent ce modèle Evolution (Ximian), SugarCRM (SugarCRM), StarOffice (Sun).

Le modèle de la souscription consistera en la mise à disposition d'un certain nombre services dont principalement l'accès aux mises à jour continues du logiciel. La souscription n'est pas un modèle de financement nouveau, nous avons détaillé ce procédé permettant de mutualiser le coût de projets communs dans la deuxième partie de notre travail [Partie II]. D'après le Directeur des Opérations de Wallix, la souscription *« est un des modèles du libre*[123]*»*. Les nouveautés du logiciel sont donc réservées aux clients de l'entreprise. Diverses firmes utilisent ce modèle : Linux Weekly News (lwn.net), Typo3, Compiere, Transgaming et Wine X (Free Software World Conference 2007: 118).

B. Les composants.

Les logiciels du type composants sont en fait des briques technologiques visant à être intégrées dans des systèmes plus ou moins larges et par conséquent à fonctionner avec d'autres logiciels libres ou non libres. Les composants peuvent être valorisés principalement selon deux modèles : (1) le dual licensing ou (2) le couplage entre une technologie de base et son complément.

Le modèle du dual licensing (Dahlander 2004; Dahlander et Magnusson 2005; Muselli 2004a; Onetti et Capobianco 2005; Välimäki 2003) dans le domaine des composants consiste à proposer une seule version du logiciel dont l'utilisateur choisira la licence en fonction de ce qu'il souhaite faire avec celui-ci. Si le logiciel est intégré dans une solution destinée à rester

[123] Directeur des Opérations, Wallix, Entretien avec l'auteur, 6 aout 2007.

libre alors il utilisera la version libre. Par contre, si l'intégration du logiciel est dans une solution non libre alors il utilisera la licence commerciale non libre. L'éditeur utilisera le caractère viral d'une licence comme élément incitant ceux ne souhaitant pas donner le code source à leurs clients à acheter une licence. D'après un G. Sadde, *« avec le modèle hybride c'est plutôt soit vous respectez les règles de l'open source et tout est gratuit et c'est un peu le langage de MySQL, soit vous ne voulez pas respecter les règles et là vous en payez le prix. C'est simple, c'est une question de philosophie : soit vous adhérez soit vous n'adhérez pas, vous avez le choix mais vous en payez le prix*[124]*... »*. Les logiciels distribués selon ce modèle sont Qt library (Trolltech), MySQL (MySQL AB).

Le modèle du couplage technologique consiste à diviser le logiciel en deux parties : d'un côté une partie générique est libre et de l'autre une partie est stratégique pour la firme donc non libre (Pal et Madanmohan 2002; West 2003). Il s'agit donc d'exploiter les avantages du libre (mutualisation de coûts, correction rapide de bugs, etc.) sur une partie non stratégique du logiciel et un modèle de valorisation directe sur la partie stratégique d'une technologie. Mac OS X (Apple) est un exemple de logiciel basé sur FreeBSD.

1.3.2. Les modèles économiques des intégrateurs.

Les intégrateurs ont deux principaux modèles économiques : (1) l'édition de distribution Linux et (2) la création de solutions spécifiques.

L'éditeur de distribution Linux (Spiller et Wichmann 2002) n'est pas réellement un éditeur mais un intégrateur de technologies libres. D'après l'ex-Directeur général de Mandriva, *« on fait un métier particulier qui est éditeur de distribution, ça veut dire qu'on n'est pas… donc si par exemple, vous allez chez MySQL, ou chez JBoss, ou chez Apache, etc. vous allez voir des développeurs d'applis etc. Nous on est des espèces d'intégrateurs, ça c'est différent, c'est un métier particulier. Notamment, on travaille avec les constructeurs, on travaille avec les autres fournisseurs de logiciels, on intègre, on planifie… C'est assez bizarre je ne connaissais pas du tout… c'est un nouveau métier ça*[125]*… »*. La valeur ajoutée de ces entreprises correspond à l'assemblage cohérent d'un ensemble de logiciels libres autour du système d'exploitation Linux. Il existe différentes distributions commerciales : Mandriva Linux (Mandriva), Ubuntu (Canonical), RHEL (Red Hat).

[124] Gerald Sadde, Avocat associé, Cabinet Roche & Associés, « OSS et distribution hybrides : réflexions juridiques », dans Conférence Ardi Numérique, L'innovation et de la compétitivité liées au développement et à l'usage de briques logicielles libres dans l'édition de logiciels, 13 novembre 2008.
[125] François Bancilhon, Ex-Directeur général, Mandriva, Entretien avec l'auteur, 22 mai 2006.

Le modèle de la création d'applications spécifiques repose la capacité d'un intégrateur à utiliser des logiciels libres et parfois même non libres pour créer une solution complète devant répondre à un cahier des charges précis. Par exemple, Thales Naval a eu pour mission de travailler sur le projet DROP (Data Resources Operating Portal), un système dématérialisé de suivi des demandes de pièces de rechange pour la Marine Nationale Française (radars, missiles, etc.). Divers logiciels libres furent utilisés : Linux, Red Hat, Apache, Struts, Tomcat, Jboss, Lucene, MySQL, Jpbm, Proxool[126]. La valeur ajoutée de l'entreprise correspond à sa capacité à assembler des briques technologiques afin de personnaliser une solution pour ses clients.

1.3.3. Les modèles économiques des fabricants de matériel.

Les fabricants de matériel ont deux types de modèles économiques : (1) vendre du matériel informatique avec des logiciels libres préinstallés et (2) vendre des produits technologiques embarquant du logiciel libre.

Le modèle de la vente de matériel informatique avec des logiciels libres préinstallés est une manière de valoriser le logiciel libre de manière indirecte (Darche, Séguin, Fermigier, Elie, Di Cosmo, et Meister 2008; Hecker 2000). Le fabricant va proposer un ordinateur en ayant au préalable installé un système d'exploitation libre ainsi qu'un ensemble d'outils correspondant à un usage générique sur le marché. La valeur ajoutée de la firme va résider dans l'étroite configuration du logiciel en fonction du matériel et la création d'une solution packagée et prête à l'emploi. Ce modèle économique combine le hardware et le software de sorte que l'économie du matériel et du logiciel soient confondus comme pendant les premiers pas de l'industrie informatique. Quelques entreprises proposent des ordinateurs avec des distributions Linux plus ou moins spécifiques : c'est notamment le cas de Dell vendant des ordinateurs avec Ubuntu préinstallé en partenariat avec Canonical ou Asus en vendant Xandros préinstallé sur ses netbook.

Le modèle de la vente de produits technologiques utilisant des logiciels libres pour fonctionner est un modèle de valorisation où il y a une imbrication très forte entre matériel et logiciel. Le logiciel libre embarqué est spécifiquement modifié et configuré pour être intégré dans un produit technologique nécessitant des fonctions spécifiques. Le logiciel libre joue un rôle plus ou moins important en fonction des produits. Henkel a mis en évidence le cas d'entreprises travaillant avec du matériel embarquant Linux (Gruber et Henkel 2006; Henkel

[126] Basé sur Grégory Lopez, Thales D3S, Thales et le Logiciel libre, 2007.

2006). D'autres acteurs utilisent du logiciel libre : certaines automobiles intègrent Java et/ou Linux ; des GPS Tomtom intègrent Linux ; certains téléphones portables Motorola intègrent Linux et Java, etc.

2. Les stratégies de verrouillage.

Dans la précédente partie, nous synthétisé les modèles économiques de l'open source en mettant en évidence le fait que la littérature avait répertorié deux types de modèles économiques : (1) les modèles ouverts : la base logicielle utilisée est libre et le résultat est aussi un logiciel libre et (2) les modèles hybrides : ces modèles conjuguent technologies libres et non libres en adoptant diverses variantes de la stratégie de double licence. Toutefois, grâce à nos investigations empiriques nous sommes en mesure de dire qu'il existe (3) une troisième catégorie de modèles économiques visant à utiliser des logiciels libres pour faire du logiciel non libre ou quasi-non libre. Il convient de noter ici que certaines licences comme la licence BSD permettent de créer un logiciel non libre à partir d'un logiciel libre (Edwards 2005: 127). Toutefois, nous allons décrire un tout autre phénomène : l'utilisation de logiciels libres sous licences virales (par exemple la GPL) pour créer un produit vendu sous licence non libre ou utiliser divers moyens technologiques pour contourner l'obligation de donner le code source des logiciels employés aux utilisateurs du produit ou service. O'Mahony dit qu'*« un vendeur commercialisant un logiciel libre qui ne fourni pas le code source ou qui ne satisfait la demande d'un client demandant le code source est une violation commune »* (O'Mahony 2003: 1187).

Malgré cette mise en évidence, la littérature ne propose pas de description concrète de ces stratégies d'appropriation. D'après nos observations empiriques des pratiques des acteurs de l'industrie du logiciel libre, bon nombre d'entreprises utilisent des moyens technologiques pour contourner les règles des licences libres. Certaines pratiques sont de purs cas de violation de licence. Par contre, d'autres cas sont beaucoup plus complexes à interpréter dans la mesure où ils ne constituent pas des violations *per se* des licences open source actuelles. Il y a d'un côté des pratiques utilisant divers procédés technologiques pour contourner les licences libres et d'un autre côté, l'évolution technologique a rendu obsolète certaines licences libres (ex : SOA, SAAS, etc.) lorsqu'elles sont appliquées dans des domaines particuliers.

Nous avons répertorié trois différentes stratégies visant à contourner les règles définies par les licences libres : (1) l'intégration d'un ou plusieurs logiciel(s) libre(s) dans du matériel, (2) le forking non libre et (3) le service en ligne.

2.1. Les stratégies d'intégration de logiciels libres dans du matériel.

2.1.1. La stratégie de TiVo.

La première manière de contourner la licence libre consiste à empêcher les modifications réalisées par les utilisateurs de fonctionner sur une machine. Le cas TiVo[127] a provoqué un véritable débat de fond au sein de la communauté du libre. TiVo est une firme américaine qui intègre des logiciels libres dans ses enregistreurs électroniques mais empêche ses utilisateurs d'utiliser des versions modifiées de son logiciel par le biais d'une protection électronique. Plus précisément, au cours du processus de fabrication, une protection est installée sur l'appareil : il s'agit de l'impression de l'empreinte numérique du logiciel préinstallé sur une puce électronique. Lors de sa mise en fonctionnement l'appareil contrôle l'empreinte numérique d'origine et celle effectivement installée sur le matériel. Lorsque l'appareil détecte une modification celui-ci refuse de fonctionner. Cette pratique est connue dans l'industrie sous le nom de *« Tivoisation »* (O'Riordan 2006).

D'après The Linux Information Project, six raisons justifient la Tivoisation : (1) protéger le système de Digital Rights Management (DRM) installé sur la machine pour contrôler le respect des droits d'auteurs des œuvres lues ; (2) empêcher la suppression d'applications envoyant des informations aux fabricants sur les habitudes de consommation des utilisateurs de manière à leur transmettre de la publicité adaptée ; (3) se conformer aux lois et directives gouvernementales et non gouvernementales en matière de sécurité ; (4) améliorer la sécurité du matériel ; (5) conserver la standardisation des produits ; (6) réduire la concurrence logicielle (The Linux Information Project 2007).

La Tivoisation a longuement été débattue par diverses parties prenantes de l'industrie du logiciel libre dont principalement la Free Software Foundation (FSF). La FSF a défini les règles des licences les plus populaires dans le monde du logiciel libre et a apporté des modifications substantielles à la version 3 de la General Public License (GPL) prévoyant entre autres l'interdiction de l'utilisation de signatures numériques dans un autre but que la sécurité.

[127] Les informations citées dans ce paragraphe ont largement basés sur les propos de Marc-Aurèle Darche, membre du conseil d'administration de l'AFUL, entretien en face à face avec l'auteur, 2 juillet 2007.

2.1.2. La stratégie de Free.

Le cas de la Freebox est un cas ayant provoqué un profond débat parmi les spécialistes de l'open source y compris au niveau juridique. La firme Free embarque des logiciels libres dans les appareils qu'elle met à disposition de ses clients moyennant des frais de location.

Les avis sont mitigés sur cette question. La Free Software Foundation (FSF) est catégorique, Free a violé la GPL en ne proposant pas aux utilisateurs d'avoir accès au code source des logiciels intégrés dans la Freebox. Pour d'autres comme l'Association Francophone des Utilisateurs de Logiciels Libres (AFUL) l'applicabilité de la GPL n'est pas évidente (Darche et al. 2008).

D'après le Délégué général à la stratégie du groupe Iliad (Free), « *Free ne viole en rien la GPL v2. Lorsqu'un produit qui utilise un soft sous GPL est vendu, le vendeur doit fournir les sources du soft GPL. Free ne vend pas la Freebox, elle est la propriété de Free, c'est un élément de terminaison de son réseau, les sources n'ont pas à être fournies […]. La Freebox est considérée comme faisant partie de l'infrastructure réseau de FREE et n'appartient jamais au client. Le logiciel GPL modifié n'est donc jamais « distribué ». Il ne sort pas du réseau de Free. En fait, ce dernier cas prête pas mal à controverse, et les GNU voudraient bien essayer de contrer ce genre de situation dans la version 3 de la GPL mais pour l'instant c'est encore assez flou. »* (Xavier Niel, dans (Afriat 2008)).

Le cas Free n'est pas simple car ce qui reste à savoir c'est premièrement la définition de la notion de distribution du point de vue de la GPL. D'après le Responsable du Centre de Compétences Open Source de Thales, *« Sur le GNU GPL. Rien n'oblige au reversement. En fait, aucune version n'oblige au reversement. Il faut rendre disponible. La pratique veut cependant que l'on reverse, donc on publie*[128]*. »* Contrairement à une idée reçue lorsqu'une modification est réalisée sur le code source d'un logiciel libre elle n'a pas à être publié mais mis à disposition des utilisateurs.

D'autre part, il faut savoir si les modifications opérées sur des logiciels libres intégrées dans une infrastructure réseau nécessitent la mise à disposition du code source des dits logiciels aux utilisateurs du réseau. La Freebox n'est pas vendue mais mise à disposition des abonnés sous la forme d'un contrat de location. La Freebox peut être vue comme une extension du réseau de Free dans le domicile de l'abonné. En outre, lorsque la Freebox est livrée à l'abonné elle ne contient pas encore le système contenant des logiciels libres mais

[128] Grégory Lopez, Responsable du Centre de Compétences Open Source, Thales D3S, entretien en ligne avec l'auteur, 8 novembre 2006.

celui-ci est téléchargé lors de la première mise en route. Il est complexe de déterminer si Free a violé la GPL mais à première vue nous ne pensons pas même si la prudence est de rigueur du fait de la complexité du dossier.

2.1.3. La stratégie des éditeurs d'UTM.

D'autres stratégies sont beaucoup plus simples à trancher car elles constituent tout bonnement et simplement des cas classiques de violation de licences libres. Le Directeur des opérations de Wallix témoigne de la situation dans le domaine de la sécurité informatique *« 90% des acteurs de la sécurité informatique, dans l'UTM [Unified Threat Management] en particulier, utilisent des logiciels libres. C'est-à-dire qu'ils intègrent des logiciels libres en tout ou partie dans leurs produits. Ils le ferment. Ils l'installent sur une machine et ils licencient. Et je pense que 90% des gens qui font ça sont dans l'illégalité au sens juridique du terme. […] Ils violent les licences clairement. […] Il y a en a à qui […] on a intenté des procès. Par exemple Fortinet a des procès pour violation de la GPL. Donc ça existe on est dans un domaine où la GPL est difficile à faire respecter… […] Dans le domaine de la sécurité on s'est aperçu qu'il y avait énormément de gens qui violaient la GPL*[129]*. »*

2.2. Les stratégies de création de logiciels non libres et services en ligne.

Dans cette section, nous évoquerons deux pratiques différentes employant des logiciels libres pour créer une solution non libre. Certains cas sont des violations pures et simples de licences libres puisque certaines firmes créent des *« forks »* non libres à partir de logiciels libres. Certaines firmes utilisent ce que les spécialistes de l'open source nomment le *« GPL loophole*[130] *»*.

2.2.1. Le cas ClosedCDConverter[131]*.*

Les extensions .daa et .uif correspondent à deux formats de fichiers respectivement générés par deux logiciels commerciaux non libres : PowerIso (.daa) et MagicIso (.uif). Ces

[129] Jean-Noël de Galzain, Directeur des Opérations, Wallix, Entretien avec l'auteur, 6 aout 2007.
[130] La *« GPL loophole »* correspond au fait que la GPL avait été créé dans un contexte technologique ne prenant pas en compte la notion d'applications en ligne. Par conséquent, la GPL ne couvre pas ce type d'utilisation et laisse la possibilité aux firmes d'exploiter cette *« faille »*.
[131] Pour des raisons de sensibilité, le nom du logiciel a été modifié.

deux logiciels permettent de créer des images de cédéroms (sous un format spécifique) pouvant être exécutés sans insérer les cédéroms originaux dans le lecteur physique.

A des fins d'interopérabilité, un développeur a réalisé du *« reverse engineering*[132] *»* sur ces deux formats et a créé deux logiciels permettant de convertir ces fichiers au format ouvert ISO. Ces deux outils open source sont DAA2ISO : convertissant les fichiers .daa en .iso et UIF2ISO : pour convertir les fichiers .uif en .iso. D'après Luigi Auriemma, le logiciel ClosedCDConverter intègre les logiciels DAA2ISO et UIF2ISO alors que : *« Le problème est que [ClosedCDConverter] n'est pas sous GPL. Il copie juste le code GPL pour gagner de l'argent du fait que les formats UIF et DAA sont très largement utilisés sur le réseau BitTorrent et que mes outils (en dehors des [logiciels] originaux PowerISO et MagicISO qui ont commencé la distribution de ce type de formats comme une stratégie marketing non éthique) sont les seuls à être capables à réaliser la conversion vers des formats comme Iso. »*

Le logiciel ClosedCDConverter est en fait distribué sous deux versions : une version de type produit d'appel distribuée en tant que freeware (logiciel gratuit non libre) qui limite la taille des fichiers à 700 mégaoctets et une autre version non bridée vendue.

Selon L. Auriemma, *« mes deux outils […] ont été copiés complètement, toutes leurs fonctions existent comme tel dans [ClosedCDConverter]*. [...] *[ClosedCDConverter] semble aussi violer la licence Qt, il utilise la version open source alors que Qt affirment que les logiciels commerciaux doivent demander une licence commerciale. »*

L'auteur du logiciel relate la manière dont il a découvert ce qu'il considère comme une violation : *« En février ou mars 2009,* […] *j'étais en train de lire quelques commentaires sur un blog où il y avait une discussion au sujet de mon outil uif2iso et j'ai vu un commentaire ressemblant à un spam à propos de ce [ClosedCDConverter] qui prétendait qu'il était capable de lire ce format. Manifestement, cela me paraissait vraiment très étrange du fait que j'ai réalisé le reverse de tous les formats et des cryptages utilisés par ce « pas si drôle » auteur de MagicISO* […] *par conséquent j'ai décidé de vérifier ça. »*

Après vérification, l'auteur s'est aperçu que ClosedCDConverter contenait les mêmes bugs que ses logiciels. D'un point vue purement et simplement factuel, la probabilité d'avoir une même erreur réalisée par deux programmes différents est très faible. En informatique l'unité élémentaire d'information s'appelle le bit celle-ci est représentée par un chiffre 0 ou 1

[132] Le *« reverse engineering »* correspond à l'action de décompiler le code objet d'un logiciel fermé pour comprendre la structuration du logiciel ou d'un format. Cette action est tolérée à des fins d'interopérabilité seulement.

(Couveignes 2009). Etant donné que la probabilité de réaliser plusieurs fois la même erreur est très faible, il est simple de détecter lorsqu'il y a copie.

Par exemple, lors de la conception de ses tables logarithmiques, Prony réalisait tous les calculs en double. Puis il vérifiait que les calculs avaient la même valeur. Lorsqu'une différence était détectée cela signifiait qu'il y avait une erreur puisque la probabilité de réaliser la même erreur deux fois de suite est très faible (de Prony 1824: 7).

Dans le cas de la comparaison de deux programmes informatique, une grande quantité d'opérations est calculée. Lorsque deux programmes réalisent les mêmes erreurs lorsqu'ils sont exécutés cela signifie très probablement qu'ils partagent une portion de code. Cela signifie très probablement qu'il a été copié.

Dans le cas de ClosedCDConverter, *« l'usage du code est simplement un complet « copier & coller » à pleines mains de TOUT mon code GPL d'une manière [tellement] honteuse et grossière que les noms de fonctions et les messages sont identiques ! »* (Auriemma 2009)

Le cas ClosedCDConverter[133] ne semble pas être un cas isolé. Plusieurs autres cas de violation de licence ont été identifiés. Par exemple, une récente affaire à opposé les développeurs de ScummVM, *« un émulateur du moteur Lucas »* (Expert n°23, University of Melbourne), et Atari. ScummVM est distribué sous licence GPL, c'est un logiciel libre particulièrement populaire que nous connaissons car celui-ci faisait parti des 25 logiciels que nous avions sélectionnés lors de notre étude sur l'innovation [Partie IV]. ScummVM a été embarqué dans les jeux Atari car la firme Atari n'a pas souhaité redévelopper une machine virtuelle pour la plate-forme Wii. D'après les membres de la communauté ScummVM, il s'agit d'un cas typique de violation de licence puisqu'aucune mention de l'utilisation de ScummVM n'est mentionnée dans les jeux distribués. Un arrangement a toutefois été trouvé entre la firme Atari et les auteurs de ScummVM (Boa 2009; ScummVM 2009).

2.2.2. Le couplage SAAS et open source.

L'industrie du logiciel est en train de passer d'une industrie de produits à une industrie de services (Sharma et al. 2002: 7). Avec la standardisation des composants et la montée des langages de programmation devenant de plus en plus basés sur des concepts, nous avons un changement de paradigme dans le développement provoquant de profondes mutations dans

[133] Nous avons appris le 5 novembre 2009 que le cas ClosedCDConverter avait été résolu : le vendeur a retiré la portion de code développée par L. Auriemma.

l'industrie informatique. De plus en plus de systèmes d'informations sont conçus à partir de *briques génériques* réalisant des opérations connues et maitrisées. Il y a de moins en moins de place pour les lignes de code et de plus en plus de fichiers de configuration de type XML[134] d'après les propos du responsable technique de Bull et Directeur Technique d'OW2 Consortium.

Deux tendances actuelles de l'industrie informatique traduisent cette évolution et donnent de plus en plus d'importance à la notion de *services* : SAAS et SOA. (1) Le *« Software As A Service »* (SAAS) est une stratégie consistant à proposer un service en ligne accessible quelque soit le système d'exploitation utilisé par le biais d'un navigateur internet. (2) La *« Services Oriented Architecture »* (SOA) est une manière de bâtir un système à partir de briques technologiques assurant des fonctions génériques nommées *« services »*.

Le couplage entre logiciel libre et SAAS provoque des synergies particulières que certains fournisseurs de services exploitent. Interrogé sur l'utilisation de logiciels libres dans le cadre de services en ligne, un avocat spécialisé en open source explique que *« dans la mesure où aujourd'hui on peut faire ce qu'on veut il n'y a aucune jurisprudence, il n'y a strictement rien sur le sujet. Je vous l'ai dit : en France, on a une jurisprudence... […] et elle ne tranche même pas ce genre de choses, les juges s'en sont bien gardés. C'est vraiment un sujet aujourd'hui tabou, on n'a pas assez de recul. Et même quand on a l'occasion d'avoir des décisions qui remontent assez haut j'ai l'impression que les juges ne veulent pas se mettre dans la logique d'analyser la validité de toute façon des licences*[135]*. »* L'intervenant du public rétorque *: « Oh, parce que comme on dit qu'aujourd'hui quand même une grosse vague de produits vont sur le mode SAAS. Logiquement on est bien si on est dans ce modèle là, on peut être en n'importe quelle licence open source et on peut distribuer au prix qu'on veut entre guillemets*[136] *! [...] »*

Par le biais du modèle du service en ligne, les logiciels ne sont parfois pas considérés comme distribués aux utilisateurs et de ce fait les firmes n'ont pas à donner le code source de leurs logiciels. Il s'agit de ce que les spécialistes nomment la *« ASP loophole*[137] *»*.

[134] XML ou eXtensible Mark-up Language, traduit en français par langage extensible de balisage. Le XML est un langage de balisage (Markup) : c'est-à-dire un langage présentant de l'information encadrée par des balises. Contrairement au HTML, le XML est un langage évolutif : il est possible de créer de nouvelles balises.

[135] Gerald Sadde, Avocat associé, Cabinet Roche & Associés, « OSS et distribution hybrides : réflexions juridiques », dans Conférence Ardi Numérique, L'innovation et de la compétitivité liées au développement et à l'usage de briques logicielles libres dans l'édition de logiciels, 13 novembre 2008.

[136] Membre du public, « OSS et distribution hybrides : réflexions juridiques », dans Conférence Ardi Numérique, L'innovation et de la compétitivité liées au développement et à l'usage de briques logicielles libres dans l'édition de logiciels, 13 novembre 2008.

[137] Application Service Provider

Nous avons débattu du cas du service en ligne avec divers spécialistes de l'open source. Selon un Open Source Architect de Thales, *« si c'est un service web sur lequel j'envoie des trames http qui contiennent une chaine de caractères et que vous me renvoyez un résultat que vous avez calculé chez vous, il n'y a pas de raison que je demande le code*[138]*. »*

D'après L. Auriemma, *« J'ai jeté un coup d'œil sur ce SAAS mais cela me semble ok avec la GPL licence du moins de ce que j'en ai compris. Cela ressemble (plus ou moins, la première chose que je peux comparer avec cela) à exécuter des applications à distance via VNC*[139]*, n'est-ce pas ? »* Cependant, il ajoute une réserve : *« dans le cas où le logiciel n'est pas exécuté sur la machine distante […] c'est distribué puisque pour être exécuté localement il [le logiciel] doit être téléchargé premièrement dans le cache ou la mémoire. »*

Diverses réactions ont été prises dans la communauté open source pour faire face à ce type de stratégies. Comme le souligne Marc-Aurèle Darche de l'AFUL[140], *« Il y a une licence qui a été créée pour empêcher ça. [...] C'est la GNU Affero Licence. [...] Le but c'était que personne ne puisse utiliser un logiciel libre derrière un serveur [sans fournir le code source aux utilisateurs]. […] C'est un détournement, je ne sais pas si on peut dire détournement, mais en tous cas c'est une évolution qui est due à l'omniprésence d'internet et des réseaux et qui n'avait pas été prévue dans la licence GPL*[141]*. »*

L'Affero General Public License (AGPL) a été créée par la Free Software Foundation (FSF) et vise à obliger les prestataires de services en ligne à fournir le code source des logiciels libres utilisés même si ces derniers ne sont pas distribués mais utilisés à distance par le biais d'un navigateur (OW2 Consortium 2008). De même la licence de Funambol, intitulée Honest Public Licence (HPL) intègre l'obligation de donner le code des logiciels libres modifiés aux utilisateurs du service (Capobianco 2006). Marc-Aurèle Darche ajoute que *« Les sociétés qui feront des choses comme ça pour moi elles ne feront pas vraiment du logiciel libre*[142]*. »*

A travers les différentes études de cas exploratoires réalisé, nous avons montré l'existence de stratégies non décrites dans la littérature et plus particulièrement dans la littérature sur les modèles économiques associés aux logiciels libres. Certaines de ces stratégies sont des violations pures et simples des licences libres. En revanche, certaines autres stratégies

[138] Open Source Architect, Thales D3S, Thales Group, Entretien en face à face avec l'auteur, 6 juin 2007.
[139] VNC est un logiciel qui permet de prendre le contrôle d'un ordinateur distant.
[140] Association Francophone des Utilisateurs de Logiciels Libres
[141] Marc-Aurèle Darche, Association Francophone des Utilisateurs de Logiciels Libres, entretien en face à face avec l'auteur, 5 aout 2007.
[142] Ibid.

exploitent les *failles* des licences libres. Ces firmes sont en quelque sorte des *« hackers »* de licences. Nous utilisons le mot *« hacker »* dans le sens proposé par un des ingénieurs de Mandriva *« un hacker c'est quelqu'un qui maitrise suffisamment un domaine pour aller au delà de l'utilisation habituelle et qui aime chercher des utilisations inhabituelles, jouer avec les limites. »* Ces firmes combinent à la fois des moyens technologiques et le contournement des règles des licences.

L'évolution technologique va certainement rendre de plus en plus complexe la distinction entre logiciel libre et fermé. La notion de *« client léger*[143] *»* est en train de devenir de plus en plus imminente dans l'industrie informatique où l'on a d'un côté une saturation des performances des microprocesseurs et d'autre part une large partie de puissance non utilisée.

Vis-à-vis de la littérature, la stratégie analysée semble constituer le contraire de ce que von Hippel et von Krogh appellent le *« private-collective model »* un modèle où *« les participants à des projets de logiciels open source utilisent leur propres ressources pour investir de manière privée dans la création d'un code de logiciel. »* (von Hippel et von Krogh 2003: 213). En effet, la stratégie du verrouillage combine l'utilisation de ressources collectives (code source de logiciels libres) et l'exploitation privée (modèle économique ad hoc) en quelque sorte nous sommes dans une *« collective-private strategy »*. Même si certains ne considèrent pas vraiment cette stratégie comme un modèle économique : *« J'aurais tendance à penser qu'il ne s'agit pas d'un modèle économique mais plutôt d'une nouvelle contrainte pour empêcher toute utilisation de code secret et donc de recentrer sur le modèle économique des services*[144]*. »*

Maintenant que nous avons traité la question des modes de valorisation des logiciels libres, nous allons nous pencher sur la stratégie open source d'un grand groupe industriel (Thales) ainsi que de la dynamique concurrentielle vue au travers de la théorie CK.

[143] En informatique, la notion de *« client léger »* consiste à mettre à disposition des applications sur des *« serveurs »* distants tout en ayant des machines dont les capacités sont volontairement limitées. C'est presque un retour aux systèmes centraux des années 60.

[144] Marc-Aurèle Darche, Association Francophone des Utilisateurs de Logiciels Libres, entretien en ligne avec l'auteur, 16 juillet 2007.

Chapitre 2. STRATEGIES INDUSTRIELLES ET CONCURRENCE.

Dans le présent chapitre, nous présenterons en premier lieu la stratégie de Thales, un groupe industriel spécialisé dans les systèmes et l'électronique dans les domaines de la défense, l'aéronautique et la sécurité [1.]. Puis nous étudierons la dynamique concurrentielle et coopérative dans l'open source à travers le prisme de la théorie CK [2.].

1. La stratégie open source d'un groupe industriel : le cas Thales.

1.1. Thales : du monde fermé à l'open source.

Thales est un groupe industriel spécialisé dans l'électronique et les systèmes et opère plus spécifiquement dans les domaines de la défense, l'aéronautique et la sécurité. Thales emploie près de 70 000 personnes à travers le monde dont plus d'un tiers sont chercheurs. Thales investit chaque année près d'un cinquième de son activité en recherche et développement. La firme est reconnue pour son expertise dans le domaine des hautes technologies. Thales partage différents laboratoires de recherche avec des institutions académiques. A titre d'exemple, Albert Fert, Directeur de l'unité mixte de recherche en physique CNRS-Thales, a remporté le Prix Nobel de Physique (2007) pour ses travaux sur la magnétorésistance.

L'engagement de Thales dans l'open source n'est pas une chose nouvelle puisque la firme a délibérément changé d'optique dans le développement en passant d'une logique totalement fermée à une logique plus ouverte. Du fait que Thales propose des systèmes dans des environnements hautement sensibles (gestion du trafic aérien, systèmes de surveillance et de sécurité, etc.), la dépendance technologique vis-à-vis d'éditeurs étrangers notamment américains posaient des problèmes importants.

Pour avoir une véritable indépendance technologique, Thales a par conséquent développé une stratégie cohérente avec cet objectif. Ainsi, Thales est un acteur de l'open source depuis plus de 15 ans. Avant de s'engager dans l'open source, Thales s'est posé trois types de questions[145] : quels étaient (1) les enjeux techniques, (2) juridiques et (3) économiques.

Du point de vue des enjeux techniques, *« c'était un passage douloureux, mais très intéressant. Notamment dans le domaine de la défense. L'open source posait problème pour*

[145] Basé sur les propos du Responsable du Centre de Compétences Open Source, Thales D3S, 23 janvier 2007.

certains composants « sensibles »[146]*. »* Thales avait une contrainte technique notamment pour le déploiement d'applications sensibles en open source. L'utilisation de l'open source dans le monde militaire, c'est une indépendance technique voulue. Bien que *« pour [tous] les ministères la criticité n'est pas la même. Pour le Minefi [Ministère de l'Economie et des Finances], c'est l'argent du citoyen. Pour la Ministère de la Défense, c'est le fantassin, comment faire pour ne pas perdre un seul fantassin. Pour la Défense, c'est un processus beaucoup plus lourd*[147]*. »*

Certains donneurs d'ordres refusaient de publier les modifications réalisées dans le code des logiciels libres utilisés dans le cadre de projets critiques. Il convient de noter toutefois que rien n'oblige à publier dans la GPL comme cela a déjà été signalé. Par conséquent, il est possible de réaliser une solution basée sur des logiciels libres (y compris sous licences virales de type GPL) et donner uniquement les modifications ainsi réalisées au client ou l'utilisateur du logiciel. Dans ce cas, le logiciel libre n'est pas un bien public[148] : c'est un logiciel conférant à un utilisateur des droits spéciaux de consultation, de modifications et de distribution. Le logiciel se rapproche de la notion de club (Buchanan, 1965).

(2) Les licences ont posé beaucoup de problèmes du point de vue juridique étant donné que *« les licences des logiciels libres ont été créées par des techniciens, « altermondialistes*[149] *». »* Les juristes de Thales ont étudié les licences et ont mis en évidence le fait que les licences libres posaient des problèmes en termes de responsabilité. Par exemple, la GPL prévoit que la responsabilité du distributeur ne peut être engagée pour tout dommage réalisé par le logiciel. Or, en droit français cette clause est nulle car il n'est pas possible de dégager sa responsabilité par le biais d'une clause : celui qui distribue le logiciel en est responsable juridiquement.

Une seconde classe de problèmes a été soulevée, elle concerne la paternité du code contribué dans l'open source. *« On s'est posé la question : est-ce c'est Thales qui donne du code ? Ou bien est-ce que c'est Grégory Lopez ? C'était un problème aussi au niveau de la DRH (Direction des Ressources Humaines), le contrat de travail est vide à ce sujet. Est-ce que l'on peut de chez nous reverser du code au nom de Thales ? Pour les techniciens, c'est la problématique de l'écosystème, s'il n'y a personne qui donne l'écosystème ne pourra pas continuer. Mais si l'on reversait du code au nom de Thales, cela signifie que la responsabilité*

[146] Grégory Lopez, Responsable du Centre de Compétences Open Source, Thales D3S, 23 janvier 2007.
[147] Ibid.
[148] L'étude de S. O'Mahony a montré que le logiciel libre n'est pas un bien public (O'Mahony, 2003 : 1195).
[149] Grégory Lopez, Responsable du Centre de Compétences Open Source, Thales D3S, 23 janvier 2007.

de Thales est engagée. Finalement, il n'y a eu aucune opposition, ni du côté « Corp. » ni du côté « Division »[150]*. »*

Et enfin, la question de l'applicabilité des licences a également été débattue. Les juristes de Thales ont souligné que *« les licences des logiciels libres sont américaines, cela signifie qu'elles n'ont aucune valeur en France en tant que licence. Pour la bonne et simple raison qu'elles ne sont pas en français ! C'est seulement une convention, un contrat*[151]*. »*

(3) D'un point de vue économique, certains ont posé la question : *« Thales veut faire de l'open source mais qu'est-ce que ça rapporte ? »* Thales créé de la valeur grâce aux logiciels libres en réalisant des systèmes sur-mesure pour les entreprises et administrations. Les logiciels libres sont utilisés comme des composants ou briques technologiques en couplant parfois technologies libres et fermées.

Thales agit aussi bien dans le domaine militaire que civil. Thales *« travaille à la fois sur de l'open source et du propriétaire. Thales travaille sur l'open source car elle considère que c'est viable d'un point de vue économique*[152]*. »* L'implication de Thales est par conséquent très pragmatique.

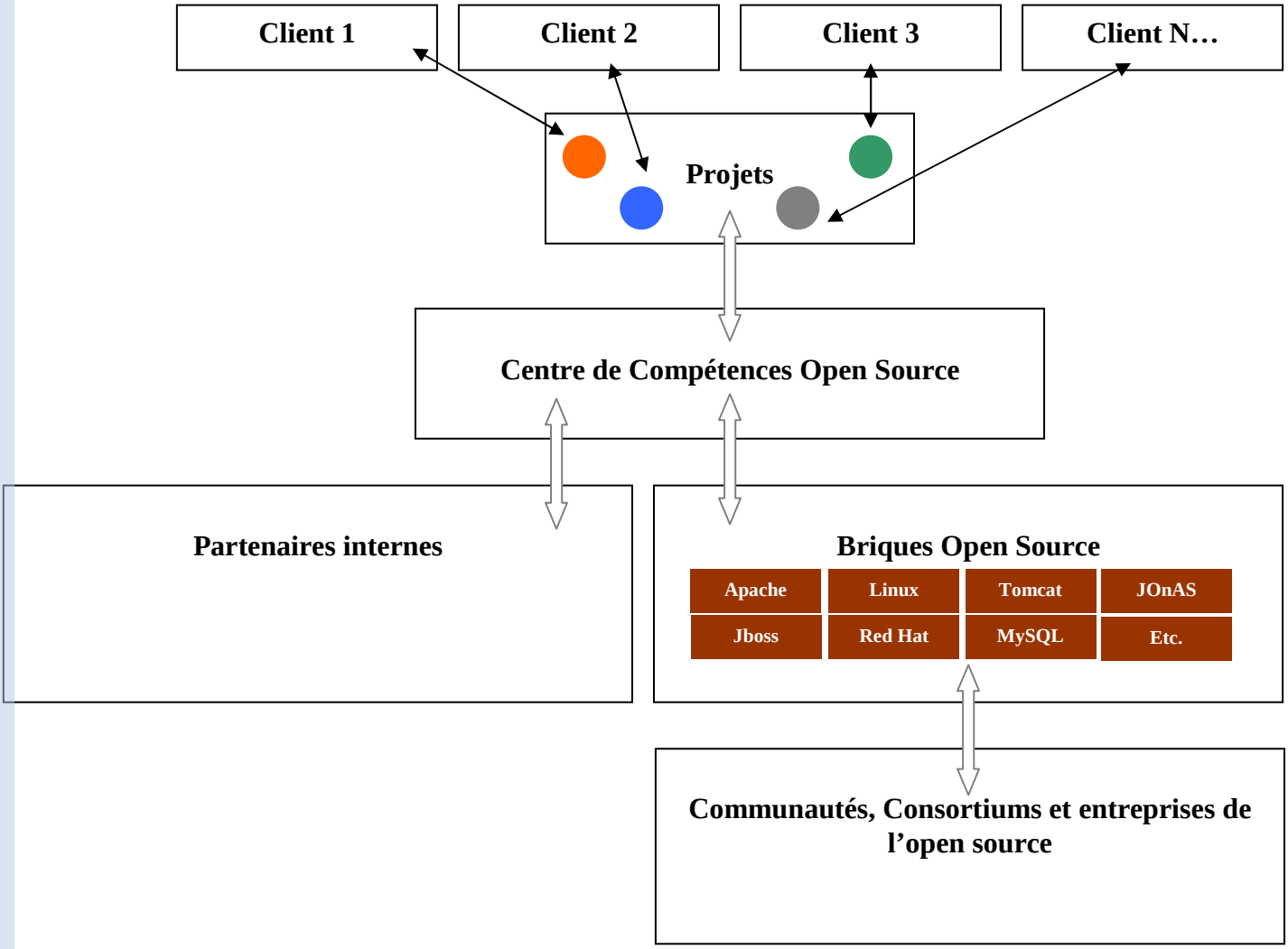
Comme le souligne le Responsable du Centre de Compétences Open Source de Thales, *« lorsque certains codes logiciels ont un enjeu technico-économique fort pour Thales alors il ne sera pas publié ! Si c'est viable pour Thales alors il ne sera pas publié, en d'autres termes si Thales est capable de maintenir cette portion ou partie du logiciel. Ce n'est pas seulement le choix de Thales, c'est aussi celui des plus grands industriels qui participent à l'open source : IBM, Bull, etc*[153]*. »* La décision de publier du code est très importante dans la mesure où parfois des portions de code sont clés pour la compétitivité de la firme. Le phénomène de la sélection des portions à publier a fait l'objet de divers travaux mettant en évidence les motivations qui sous-tendent la publication ou la non-publication du code source (Gruber et Henkel 2006; Henkel 2006, 2007).

En 2006, la création du Centre de Compétences Open Source (CCOS) est un pas décisif dans le changement de stratégie de Thales. La création du CCOS *« répond à l'anticipation sur l'évolution des méthodes de développement, d'une logique propriétaire, vers une logique ouverte, appliquée à l'informatique de services. Son rôle est d'explorer cette évolution et de construire un réseau d'alliance stratégique de moyennes et grosses entreprises*[154]*. »*

[150] Grégory Lopez, Responsable du Centre de Compétences Open Source, Thales D3S, 23 janvier 2007.
[151] Ibid.
[152] Ibid.
[153] Ibid.
[154] Viet Ha, Open Source Architect, Thales D3S, entretien en ligne avec l'auteur, 21 aout 2007.

Le CCOS est un service de Thales faisant l'interface entre les projets commandités par des clients (Marine Nationale, Minefi, etc.), les composants open source développés dans le cadre de communautés et de consortiums industriels (OW2, Eclipse, Apache, Nagios, etc.) et des partenaires. Ci-dessous un schéma propose une vision simplifiée du rôle du CCOS.

Figure 26 : Le rôle du Centre de Compétences Open Source[155].



Le CCOS assure la pérennité de l'architecture logicielle et des applications, cela signifie que les composants open source peuvent évoluer ou même être remplacés en fonction de trois éléments : (1) le temps, (2) les évolutions technologiques et des standards, (3) des besoins métiers et clients ceci sans remettre en question le fonctionnement du système conçu par Thales[156].

[155] Schéma inspiré de Grégory Lopez, Responsable du Centre de Compétences Open Source, Thales D3S, Thales et le Logiciel libre, 2007, page 21.
[156] Basé sur Grégory Lopez, *ibid.*, page 11.

D'après nos investigations, le positionnement de Thales dans le domaine de l'open source est justifié par des motivations économiques. Les logiciels libres permettent une réutilisation de composants existants qui accélèrent les cycles de développement. L'industrie du logiciel devient de plus en plus orientée vers les services, surtout dans le domaine des grands systèmes d'information où les clients ne souhaitent plus être dépendant d'un seul interlocuteur pour le maintien de leur système.

Il y a également des motivations technologiques qui justifient le recours aux logiciels libres : la recherche d'une indépendance technologique. Thales a une taille lui permettant de fédérer plusieurs projets industriels autour de briques logicielles communes.

Enfin, le logiciel libre est utilisé dans des contextes sensibles car *« le modèle de la communauté est beaucoup plus réactif. Par exemple, le Minefi*[157] *a eu un bug bloquant sur son site de paiement des impôts avec Jboss*[158]*. Le problème a été résolu en 9 heures ! Il y a eu une version officielle qui prenait le correctif en 9 heures, c'est du jamais vu. Même les grands éditeurs n'ont pas cette réactivité. Les communautés sont beaucoup plus réactives*[159]*. »* De même, *« dans l'open source on a des capacités de débugge très grande bien meilleure que dans le domaine du logiciel propriétaire. Cela améliore la qualité du logiciel*[160]*. »*

1.2. Thales et l'open source.

Thales a différentes initiatives et est impliqué dans différentes organisations visant à promouvoir les logiciels libres et les standards ouverts. Nous avons fait le choix de classer ces activités en deux catégories : les activités non stratégiques et les stratégiques pour le groupe.

1.2.1. Les activités open source non stratégiques pour Thales.

Thales est membre de l'April, l'ODF Alliance et Eclipse. (1) L'APRIL (Association Pour la Promotion et la Recherche en Informatique Libre) est une association visant à promouvoir les logiciels libres et les standards ouverts dans toute l'industrie informatique. L'APRIL a une vision du logiciel libre très proche de la FSF (Free Software Foundation) donnant une importance particulière à la philosophie du logiciel libre. Comme G. Lopez le souligne, *«*

[157] Ministère de l'Economie et des Finances.
[158] JBoss est un serveur d'application open source.
[159] Grégory Lopez, Responsable du Centre de Compétences Open Source, Thales D3S, 23 janvier 2007.
[160] Serge Druais, Directeur de la Recherche et de l'Innovation, Thales D3S, entretien téléphonique avec l'auteur, 5 aout 2009.

nous pensons que L'APRIL joue un rôle extrêmement important dans l'écosystème Logiciel Libre en France. Ses principales actions, défense et promotion, sont pour nous essentielles. Même si nous ne sommes pas toujours en parfait accord avec certaines idées, adhérer à l'APRIL nous permet nous aussi de participer et de faire avancer le débat, de préserver un modèle technique et économique viable et fiable. Par exemple, nous qui faisons massivement la promotion d'une indépendance technologique Européenne, les actions en faveur des standards ouverts de l'APRIL sont un appui formidable[161]*. »* Plus concrètement, *« Thales est à l'APRIL pour : participer à l'effort libre, être reconnu par les autres et faire du Networking*[162]*. »*

(2) L'ODF Alliance (Open Document Format Alliance) est un consortium visant à promouvoir un format de document ouvert ayant pour but l'interopérabilité et la bonne conservation des documents. L'ODF Alliance est typiquement une organisation ayant pour but la restauration de la concurrence sur le marché de la bureautique où les formats de documents de Microsoft se sont littéralement imposés en tant que standards de fait. En outre, l'ODF est concurrent du format Open XML promu par Microsoft certifié ISO en 2007 (International Organization for Standardization (ISO) 2008).

La défense de l'ODF s'inscrit dans la stratégie globale d'indépendance technologique défendue par Thales mais elle n'est pas stratégique pour le groupe dans la mesure où l'enjeu des formats de documents n'est pas clé pour la compétitivité de Thales contrairement à Sun Microsystems et IBM ayant leur propre suite bureautique, respectivement StarOffice et Lotus Symphony. Par conséquent la participation de Thales dans l'ODF Alliance est assez faible.

(3) Eclipse est une communauté rassemblant plus de quatre-vingt entreprises (Rational Software (IBM), Oracle, CA Inc., SAP, etc.) ayant pour objectif de fournir des plates-formes et des applications ouvertes pour la production de logiciels. Eclipse est un consortium dont le leadership est plutôt nord-américain et plus précisément celui d'IBM à travers sa division Rational Software. Selon G. Lopez, *« Eclipse faut pas se leurrer, c'est IBM ! »* La participation de Thales à Eclipse est beaucoup moins importante pour la stratégie du groupe : Thales n'est que membre associé.

1.2.2. Les activités open source stratégiques pour Thales.

[161] Grégory Lopez, Responsable du Centre de Compétences Open Source, Thales D3S, dans *Témoignages sur l'April*, April, décembre 2006.
[162] Grégory Lopez, Responsable du Centre de Compétences Open Source, Thales D3S, entretien en face à face avec l'auteur, 23 janvier 2007.

Thales a différentes initiatives open source aussi bien interne qu'externe auxquelles la firme participe activement et dont les enjeux sont beaucoup plus importants pour le groupe.

A. Initiatives open source internes : POSSIS et Thalix.

Au niveau interne Thales a mis en place une initiative nommée POSSIS. (1) POSSIS est un programme interne à Thales ayant pour but la création d'une plateforme ouverte orientée services pour les systèmes d'information critiques. POSSIS est une initiative open source interne rendue possible par le fait que Thales est un groupe d'une taille importante (près de 70 000 collaborateurs). POSSIS comprend plus de 40 composants libres utilisés dans des contextes industriels.

(2) D'autre part, Thales a développé sa propre distribution Linux basée sur RHEL (Red Hat Entreprise Linux) nommée *Thalix*. Thalix est une distribution dédiée à la gestion du trafic aérien, c'est *« une distribution basée sur Redhat, construite par Thales Air Traffic Management*[163]*. »* Thalix est une distribution allégée en paquets[164] afin d'être le plus stable possible. Thalix *« est une version « strip-down », dont le système de mise à jour a été enlevé, pour garantir l'unicité de la plateforme dans les centres de contrôle aérien*[165]*. »*

B. Activités externes : initiatives partiellement open source.

(1) Au niveau externe, Thales participe à ARTEMIS (Advanced Research and Technology in Embedded Intelligence and Systems), une association de droit néerlandais regroupant des grandes entreprises, des PME et des centres de recherche. ARTEMIS est un consortium dédié aux systèmes embarqués. Thales fait plus particulièrement parti du comité de direction d'ARTEMIS.

(2) NESSI (Networked European Software & Service Initiative) est un consortium présidé par Thales. A l'origine, il a été fondé par 22 grandes entreprises européennes. Aujourd'hui, NESSI rassemble plus de 400 organisations de l'industrie informatique et des télécommunications. NESSI comprend une moitié d'acteurs de la recherche académique, un quart de grandes entreprises et un quart de PME. NESSI est dédié au développement d'une industrie de services européenne. Les logiciels libres sont au cœur de cette initiative car ces derniers permettent de développer des briques génériques pouvant être réutilisées dans

[163] Viet Ha, Open Source Architect, Thales D3S, entretien en ligne avec l'auteur, 21 aout 2007.
[164] Les paquets sont les programmes compris dans une distribution Linux.
[165] Ibid.

différents contextes. NESSI vise également à promouvoir des standards ouverts assurant une compatibilité des différents services. NESSI est autofinancé par les partenaires.

C. La participation de Thales au projet européen QualiPSo.

QualiPSo est un projet européen visant à améliorer la qualité des logiciels libres. Plus spécifiquement, QualiPSo envisage la définition et l'implémentation *« des technologies, processus et politiques visant à faciliter le développement et l'usage de composants open source, avec le même niveau de confiance traditionnellement proposé par les logiciels propriétaires »* (QualiPSo 2007). Il s'agit d'un projet financé par la Commission Européenne à hauteur de 22 millions d'euros. QualiPSo regroupe 21 organisations (Thales, European Dynamics, la Gendarmerie Nationale, etc.).

D'après le responsable de l'activité *« Factory »* de QualiPSo, *« l'open source pose un problème pour les industriels à savoir : il y a des softs vraiment très bien mais il n'y a pas de support, il n'y a pas de qualité, on ne sait pas ce que ça vaut. Qu'est-ce qui se passe si dans trois ans on s'arrête de travailler dessus ? Il y a un gros problème en fait et QualiPSo essaye de répondre à cette situation en mettant en place d'une part des solutions logicielles et d'autre part des solutions légales et organisationnelles. Du point de vue logiciel qu'est-ce que l'on peut faire ? On peut essayer de nourrir les communautés en leur fournissant un environnement informatique sur lequel ils vont se trouver à l'aise, sur lequel ils vont pouvoir, échanger des informations, donc l'idée de collaboration*[166]*. »* D'autre part, selon le Responsable du CCOS de Thales, *« Dans le monde des éditeurs, quand on a un contrat avec IBM et que l'on a un problème on peut se retourner sur IBM. Mais quand c'est une communauté c'est différent ! Vers qui se retourner*[167] *? »*

Nous voyons bien que la qualité des logiciels libres n'est pas aussi clairement définie que le prétendent certains auteurs de la littérature (Raymond 1999, 2000; Spaeth, von Krogh, Stuermer, et Haefliger 2007: 2). Comme le souligne Fuggetta, *« En général, il est difficile d'affirmer qu'il y a une relation causale entre le fait que le logiciel soit open source et son efficience, qualité et valeur »* (Fuggetta 2003: 87). QualiPSo est donc la preuve qu'il y a un réel débat autour de la qualité des logiciels libres. Selon le responsable de l'activité centres de compétences, *« la qualité d'un logiciel libre ne se mesure pas seulement à la qualité du code. […] Il y a deux choses à considérer : il y a des critères techniques et des critères non*

[166] Viet Ha, Open Source Architect, Thales D3S, entretien avec l'auteur, 6 juin 2007.
[167] Grégory Lopez, Responsable du Centre de Compétences Open source, Thales D3S, entretien avec l'auteur, 23 janvier 2007.

techniques. Dans la partie technique on peut effectivement vérifier la conformité aux standards, la compatibilité avec d'autres logiciels, est-ce que le code est écrit proprement, etc. Après il y a tous ces critères non techniques au moins aussi importants dans le cadre d'un logiciel libre. Par exemple on peut avoir un code de très très bonne qualité mais s'il n'y a jamais de documentation, si quand vous posez une question sur la mailing list jamais personne ne vous répond, ce genre de choses. En ce qui concerne le logiciel libre, il faut vraiment prendre en compte les deux aspects. Dans QualiPSo, les deux sont très liés, on a une activité qui s'intéresse aux facteurs de qualité du logiciel libre, qui développe des outils pour analyser la qualité technique et aussi des métriques pour évaluer les aspects non techniques[168]*. »*

D'autre part, *« l'un des mots clés est « métrique », comment définit-on les métriques pour mesurer cette qualité ? Il y a un certain nombre de projets qui visent à définir un certain nombre de métriques de qualité. Les métriques ça peut être parcourir les forums et regarder en combien de temps les gens ont une réponse à leur problème, combien il y a de bugs qui sont postés, quelle est la vie qui a autour du projet, il y a des critères techniques. Est-ce qu'il y a de la documentation ? Est-elle régulièrement mise à jour*[169] *? Etc. »*

Ce projet européen peut être vu comme une tentative de *corporation* (au sens français du terme). En d'autres termes, ce qui est recherché c'est créer une normalisation de la qualité des logiciels libres afin qu'ils aient un niveau de qualité similaire aux logiciels fermés.

QualiPSo est découpé en dix activités. Thales est officiellement impliqué dans trois activités : l'activité 1 porte sur les aspects juridiques de l'open source ; l'activité 7 concerne la conception d'une *Factory* d'outils au service des développeurs open source et l'activité 8 se préoccupe de la pérennité du projet par le biais de la création de centres de compétences. Nous avons seulement étudié les activités 7 et 8 étant donné que l'activité portant sur les aspects légaux est traitée par le département juridique de Thales Group.

Pour l'activité Factory, QualiPSo proposera une nouvelle génération de forge comprenant divers outils que les forges existantes ne proposent pas. Dans les forges traditionnelles (SourceForge, GeForge, etc.), il y a très peu d'outils de collaboration. Ces forges se limitent souvent à un espace où les développeurs déposent du code. L'idée de la Factory est donc bien différente puisque celle-ci vise à *« à améliorer les facteurs de qualité dans le processus de développement continu de façon générale et tout ça et l'idée c'est de l'exploiter ensuite d'en faire la promotion à l'extérieur par un réseau de centres de compétences open source : les*

[168] Julie Marguerite, Open Source Architect, Thales D3S, entretien avec l'auteur, 6 juin 2007.
[169] Viet Ha, Open Source Architect, Thales D3S, entretien avec l'auteur, 6 juin 2007.

centres de compétences de QualiPSo. L'idée dans le projet c'est d'en avoir six : quatre en Europe, un au Brésil puis un en Chine. Puis là l'objectif, c'est de faire la promotion, c'est de vendre du service sur les technologies développées dans QualiPSo[170]*. »*

QualiPSo ce n'est pas seulement un projet européen, il y a aussi *« un objectif de pérennité hors du cadre du projet. Puis après les centres de compétences sont sensés s'auto-suffire et exister de façon autonome. Pour ça ils vendront du service sur les technologies open source mais ils vendront des services associés aux technologies*[171]*. »*

D. Les enjeux et le rôle de Thales dans le consortium OW2.

Selon le Responsable du CCOS, *« plus on fait de l'open source, plus on est crédible dans le monde de l'open source. L'implication dans les communautés devient un atout [concurrentiel], il y a une preuve de la compétence : c'est la présence dans des communautés*[172]*. »* En lisant ces propos, nous comprenons pourquoi Thales est présent dans diverses communautés open source dont OW2 Consortium (OW2).

Thales ne bâti pas sa légitimité sur l'édition de produits logiciels comme d'autres acteurs de l'informatique mais plutôt car *« Thales est capable de faire des systèmes de surveillance pour la police, des systèmes de gestion du trafic, etc. On ne choisit pas Thales car on s'appelle IBM Websphere*[173]*. »* La participation de Thales dans OW2 est donc clairement dans *« une volonté d'indépendance technologique*[174]*. »* L'enjeu de la participation de Thales est également de *« positionner Thales comme un acteur crédible dans le segment Open Source*[175]*. »*

Selon le représentant du board d'OW2, *« Historiquement, on utilisait OW2 comme vecteur pour disséminer. On a fait de la R&D en génie logiciel que Thales va mettre en open source. Libérer des logiciels dans lesquels il n'y a pas de valeur pour Thales. OW2 c'est un socle technologique. L'intérêt, c'est le partage avec d'autres qui font aussi vivre le logiciel. Ce n'est pas qu'avec des couches intermédiaires qu'on fait de la valeur. D'autres personnes*

[170] Julie Marguerite, Open Source Architect, Thales D3S, entretien avec l'auteur, 6 juin 2007.
[171] Ibid.
[172] Grégory Lopez, Responsable du Centre de Compétences Open Source, Thales D3S, entretien avec l'auteur, 23 janvier 2007.
[173] Serge Druais, Directeur de la Recherche et de l'Innovation, Thales D3S, entretien téléphonique avec l'auteur, 5 aout 2009.
[174] Ibid.
[175] Grégory Lopez, Responsable du Centre de Compétences Open Source, Thales D3S, entretien en ligne avec l'auteur, 23 aout 2009.

peuvent nous signaler des bugs qu'il nous reste plus qu'à corriger, c'est du travail qu'on n'a pas à faire : identifier les bugs.[176]*. »*

Concernant le rôle de Thales dans OW2, Thales participe à tous les niveaux dans le consortium. En effet, Thales est *« membre du Board, du chapitre local Europe, du management office et du Technical comittee*[177]*. »*. (1) Au niveau des conseils, Thales participe à l'Ecosystem Council et au Technology Council. (2) Au niveau stratégique, Thales est strategic member d'OW2, la firme est un des acteurs clé du consortium et a un poids important sur les orientations de celui-ci. Elle dispose d'un représentant au comité de direction d'OW2. (3) Au niveau des activités, Thales est leader de l'e-Gov Initiative. Cette initiative est une initiative OW2 ayant pour but de sélectionner des composants open source d'OW2 et hors du consortium pouvant être mobilisés dans le cadre de projets de systèmes d'information pour des administrations publiques en Europe et en Chine. Selon G. Lopez, *« l'e-gouvernement c'est l'utilisation des technologies de l'information et de la communication, combinées avec le changement organisationnel et de nouvelles compétences pour améliorer les services publics, augmenter la participation et améliorer les politiques publiques*[178]*. »*

Le cas Thales décrit comment une entreprise agissant dans un domaine exigeant en matière de qualité et de sécurité, a changé d'optique de développement en intégrant des technologies libres et en partageant des technologies non critiques pour la compétitivité du groupe. Comme le souligne un membre d'OW2, *« je pense que l'open-source a un pouvoir inné de transformer les organisations même les plus fermées et les plus policées : une fois qu'il y a fait son entrée, les choses changent, et çà échappe à tout contrôle (la remarque vaut pour Thalès... comme pour la Chine entière*[179]*). »*

La stratégie de Thales prouve bien que la sécurité n'est pas forcément une question d'utilisation de logiciels de type *boîte noire* dont on ne connaît pas le contenu. Selon L. Laudinet, *« la plupart des logiciels propriétaires contiennent des failles de sécurité volontaires ou non)*[180]*. »*

[176] Serge Druais, Directeur de la Recherche et de l'Innovation, Thales D3S, entretien téléphonique avec l'auteur, 5 aout 2009.
[177] Grégory Lopez, Responsable du Centre de Compétences Open Source, Thales D3S, entretien en ligne avec l'auteur, 23 aout 2009.
[178] Grégory Lopez, Responsable du Centre de Compétences Open Source, Thales D3S, eGov Initiative Presentation and status THALES, eGov meeting, 28th May 2007, Beijing.
[179] Pierre-Yves Gibello, Experlog, entretien en ligne avec l'auteur, 31 juillet 2009.
[180] Laurent Laudinet, Open Source Architect, Thales D3S, Rencontres Mondiales du Logiciel Libre, 2007.

Après avoir présenté la stratégie open source de Thales, nous vous proposons de nous pencher sur la question de la dynamique concurrentielle vue à travers la théorie CK.

2. La dynamique concurrentielle et coopérative dans l'open source.

2.1. L'imbrication des connaissances dans l'open source : généralités.

Classiquement, certains auteurs et autres partisans du logiciel libre défendent l'idée que du fait qu'un logiciel est libre tout le monde est en mesure d'apporter des modifications sur le logiciel et à son tour distribuer le logiciel résultant des modifications[181]. D'autres ajoutent que dans le domaine professionnel n'importe quelle firme peut proposer des services et des développements spécifiques pour n'importe quel logiciel libre étant donné que le code source de ces logiciels est mis à disposition soit des utilisateurs (c'est l'obligation de la GPL) soit de tous par le biais d'une URL publique (c'est en principe le comportement socialement attendu).

En réalité, von Hippel et von Krogh ont mis en évidence le fait que le free-riding n'était pas un problème dans l'open source car les développeurs avaient des bénéfices en termes de connaissances (von Hippel et von Krogh 2003). En s'appuyant sur ces travaux nous pensons qu'il s'agit d'une vision *simpliste* de la dynamique concurrentielle dans l'open source. Nous verrons que le fait qu'un logiciel soit libre ne signifie pas que tout le monde peut l'améliorer et être concurrentiel en termes de services associés. Il s'agit d'un *véritable mythe* qu'il est nécessaire de briser.

D'un point de vue caricatural, l'économie du logiciel fermé est basée sur une logique de verrouillage des connaissances produites et l'économie du logiciel libre est basée sur une logique d'ouverture des connaissances. Certains auteurs voient l'open source comme un cas extrême d'open innovation (Dahlander et Wallin 2006: 1243).

Lors de nos investigations sur le terrain, nous avons plusieurs fois vu que la maîtrise du code source d'un logiciel n'est pas une chose simple et que cela nécessite d'avoir des compétences adéquates. Trois témoignages nous ont amenés à creuser l'idée qu'il y avait un couplage entre code source et connaissances.

(1) Le premier témoignage porte sur StarOffice. A l'époque où Sun Microsystems a racheté StarDivision (SD), les développeurs de la SD *« étaient au lancement de OOo les seuls*

[181] Par exemple, Johnson prétend que le code source d'un logiciel est disponible, un développeur compétent peut réaliser des modifications, des extensions, etc. (Johnson 2002: 638).

à pouvoir lire le code de StarOffice, peu documenté à l'époque. Leur implication dans OOo était donc primordiale[182]*. »*

(2) Le second porte sur Java. Lors d'une entrevue, un spécialiste des technologies de Sun Microsystems expliquait la libération de Java : *« comme l'a souligné le responsable du projet Geronimo, « il y a des milliers de jours homme dans la Java Standard Edition ». Aujourd'hui les gens capables d'entrer dans un projet pareil, comme dans Open Solaris d'ailleurs, et de faire des modifications ils sont extrêmement peu nombreux*[183]*… »*

(2) Le troisième porte sur le cas Exo Platform et Unbreakable Linux. D'après G. Lopez, Exo Platform a montré que certaines communautés sont organisées pour faire face à une logique financière. Exo Platform a un modèle économique couplant logiciel libre et expertise associée à cette technologie[184]. En 2006, Oracle déclaraient qu'ils proposaient une nouvelle version de RHEL. Selon, G. Lopez, *« Oracle ne pourra pas répondre seul à des bugs de niveau 3 [niveau élevé] qui implique une excellente connaissance du code de Redhat. Si ils ont annoncé ça, c'est qu'ils ont passé un accord cadre avec Red Hat prévoyant l'accès au Lab de Red Hat en cas de besoin. Nous sommes bien placés pour dire ça, car nous assurons aussi le support de Red Hat et nous avons aussi un accord cadre avec Red Hat [pour THALIX] prévoyant l'accès au Lab si besoin est*[185]*… »*

Nous vous proposons d'étudier les mécanismes développés par les firmes de l'open source pour contrebalancer les effets de la libération d'une technologie. Pour ce faire, nous emprunterons le diptyque concepts (C) / connaissances (K) proposé par la théorie CK (Hatchuel et al. 2002; Le Masson, Weil, et Hatchuel 2006). Cette théorie propose de décomposer la construction de produit en deux espaces interdépendants : l'espace des concepts et l'espace des connaissances. Une connaissance désigne une proposition disposant d'un statut logique. Un concept est une nouvelle proposition qui n'a pas forcément de statut logique (Kazakci 2007: 37-38).

[182] Membre, OpenOffice.org, Entretien en ligne avec l'auteur.
[183] Technology Advisor, Sun Microsystems, Entretien en face à face avec l'auteur, 15 janvier 2007.
[184] Basé sur les propos de Grégory Lopez, Responsable du Centre de Compétences Open Source, Thales D3S, entretien en face à face avec l'auteur, 23 janvier 2007.
[185] Ibid.

2.2. L'imbrication des connaissances vue sur le terrain.

2.2.1. Le cas THALIX.

Dans certains cas la firme éditrice (E_1) n'est pas en mesure de répondre au marché auquel (E_2) souhaite répondre en intégrant un composant (nous le nommerons « SC »). Il est possible de relire le cas Thalix, en se focalisant sur la dynamique des connaissances. Red Hat, l'éditeur de Red Hat Entreprise Linux (RHEL) n'avait pas les compétences internes lui permettant de créer un système de gestion du trafic aérien. Alors que Thales a bâti une longue expertise dans le domaine aéronautique en intégrant des technologies fermées.

Un Open Source Architect de Thales nous expliquait que *« historiquement, les serveurs utilisés avant l'arrivée de ces machines sous Thalix, étaient des « Stratus[186] », dont le matériel et le système d'exploitation étaient contrôlés par un fournisseur unique. Il en découle les problèmes de compétence disponible, de réactivité de la hotline unique, en particulier face à des pannes matérielles dont quasiment personne n'avait l'expérience[187]. »* Thalix est une distribution Linux spécifique très différente de ce qui est demandé sur le marché des systèmes d'exploitation.

Pour la gestion du trafic aérien, *« la stabilité de Thalix étant le critère principal, c'est une distribution qui maintient un retard de sécurité d'environ trois ans sur l'existant logiciel courant. Il en découle une compatibilité matérielle effective avec des ordinateurs contemporains de sa conception. »* Thalix *« n'est donc pas une distribution grand public, elle n'est donc pas distribuée comme telle. Elle est en revanche mise en avant dans les réponses à appel d'offre qui requièrent une stabilité exemplaire des serveurs (quand le matériel n'a pas de contrainte de modernisme particulier). »* En 2009, un avion sur deux est géré par le biais des systèmes de gestion du trafic aérien de Thales[188].

Red Hat était donc dans l'impossibilité de créer ce système de gestion du trafic aérien car la firme ne disposait pas des compétences nécessaires. Thales a utilisé des connaissances externes K_X que Red Hat ne disposait pas. Les K_X ont permis à Thales d'acquérir un revenu spécifique lié à la fois au code source de RHEL (SC_{RHEL}) et ces K_X.

Dans le cas Thalix, il n'est pas vraiment possible de parler de concurrence pour Red Hat car elle n'était pas capable de répondre à cette demande. Au lieu de cela, Red Hat a réussi à

[186] Stratus Technologies propose des serveurs et systèmes pour les environnements nécessitant une haute fiabilité. http://www.stratus.com/
[187] Viet Ha, Open Source Architect, Thales D3S, entretien en ligne avec l'auteur, 21 aout 2007.
[188] Thales Group, THALES Presence in Air Traffic Management, Janvier 2009.

bénéficier d'une rente (P_3) qu'elle n'aurait pas eu autrement. Thales a en effet passé un accord cadre donnant accès au laboratoire Red Hat en cas de bug de niveau 3.

Red Hat bénéficie donc de :

$$[P_2 (SC/K_2) + P_3 (Thales)]$$

Thales bénéficie de[189] :

$$[P1 (SC/K_1K_X) – P_3]$$

2.2.2. Le cas JORAM.

Lors d'une conférence professionnelle dédiée aux technologies open source un débat très intéressant a émergé entre le Directeur général de ScalAgent Distributed Technologies (éditeur de JORAM) et un intervenant du public. Dans ce cas aussi nous montrerons qu'il y a une imbrication entre le code source et les connaissances nécessaires à l'exploitation marchande du logiciel.

JORAM est un logiciel complexe dont *« la base technologique est une plateforme à agents distribuée, développée dans un centre de recherche commun Bull - Inria - Universités de Grenoble. Lorsque la norme JMS est sortie, nous avons réalisé une implantation sur notre base technologique[190].* » Pour ScalAgent *« la décision open source a été prise par les créateurs [de JORAM] […] Elle se justifie de par la taille des concurrents (IBM, Oracle). »* Dans ce cas, le choix de l'open source repose sur une réelle stratégie économique. Le modèle économique associé à JORAM repose essentiellement sur du *« service de différents type : architecture, conseil, support, extensions spécifiques et plus largement vente de notre expertise en architecture distribuée. »*

Nous vous proposons quelques passages de ce débat[191]. D'après l'intervenant du public : *« Quel est le risque si demain vous avez un CapGemini ou un Sopra qui veut se mettre sur votre métier ? Il y a un moment pour prendre de l'expérience j'imagine car il s'agit d'un produit technique complexe je l'imagine. Mais quel est le niveau de risque par rapport à une structure, vous dites encore modeste [...] et un Sopra pour prendre un exemple au hasard...*

[189] P_3 = le montant de l'accord cadre passé entre Red Hat et Thales.
[190] Citations basées sur les propos de Serge Lacourte, Directeur Général, ScalAgent Technologies, Entretien en ligne avec l'auteur, 17 juillet 2009.
[191] L'ensemble des citations de Serge Lacourte et de l'intervenant public sont issues de la conférence, Ardi Numérique, du 13 novembre 2008, intitulée *« l'innovation et de la compétitivité liées au développement et à l'usage de briques logicielles libres dans l'édition de logiciels »*.

qui viendrait mettre 40 bonhommes sur l'activité et devenir votre unique intégrateur de votre unique solution. »

Serge Lacourte (Directeur Général, Scalagent Distributed Technologies) : *« L'unique intégrateur je ne pense pas puisque c'est déjà le cas il y a beaucoup d'intégrateurs qui utilisent JORAM et qui le placent. Par exemple, on a appris incidemment que JORAM a été utilisé pour traiter les messages SMS d'une émission […] »*

Intervenant public : *« Ce que je veux dire c'est si on a quelqu'un qui a une force marketing énorme et la capacité à apprendre vite sur le produit. Est-ce que ce n'est pas un risque majeur ? Enfin JORAM ou un autre produit, peu importe... Est-ce que ce n'est pas un risque majeur du libre ? Quand on est une petite équipe à côté... »*

Serge Lacourte : *« Pour moi, il y a une partie où il n'y a pas de risque déjà puisque c'est une part de marché qu'on ne prend pas : c'est toute cette la partie intégration. […] Après c'est sur cette partie : services spécifiques, services intégrateur, c'est possible que quelqu'un qui ne connait pas la techno décide... qui la prenne comme vous le dites, décide de prendre également ces services.*

J'ai l'impression que la problématique est la même. [...] D'abord, [...] je pense qu'il va prendre du temps avant de maîtriser la techno parce que malgré tout il y a une innovation assez importante à l'intérieur donc avant qu'il rentre dedans c'est un peu compliqué.

Je vois déjà sur la partie architecture, dans ce que je vous ai signalé c'est du réel. […] Certains intégrateurs prennent le contrat, font une architecture qui correspond pas... donc vous voyez avant qu'ils comprennent et qu'ils fassent les bonnes architectures, je pense qu'on a le temps ! Rien que cette première étape, si vous voulez ils ne savent pas la faire. Donc avant qu'ils prennent vraiment la connaissance du produit qu'ils soient capables de faire les extensions, qu'ils soient capables de comprendre le code et de debugger... là je pense qu'on a vraiment le temps. [...] enfin de trouver un bug dans ce logiciel ce n'est quand même pas si évident que ça. Ca demande quand même un investissement important.

Je pense que l'intégrateur a de toute façon une part de son métier : c'est la mise en œuvre, c'est du dimensionnement, ça va être une expérience dans tout un tas de contextes différents ça c'est le métier de l'intégrateur. […] Je pense qu'au départ il va se concentrer là dessus et c'est là dessus qu'il veut faire sa valeur ajoutée. Qu'il essaye de prendre en plus l'autre... Il n'a pas besoin d'investir... d'abord il n'investira pas pour assurer ça. »

Dans le cas JORAM, nous nous apercevons qu'il y a un phénomène similaire de dynamique des connaissances et de coopération entre une firme utilisatrice de JORAM (un intégrateur) et l'éditeur de JORAM (ScalAgent). La mise à disposition de JORAM en open

source permet à ScalAgent de dégager des revenus qu'elle n'aurait pas été en mesure de générer du fait de sa taille puisque JORAM est utilisé dans des grands projets d'intégration.

ScalAgent bénéficie donc de :

$$[P_2 (SC/K_2) + P_3 (\text{Intégrateur})]$$

L'intégrateur bénéficie de[192] :

$$[P_1 (SC/K_1K_X) - P_3]$$

2.3. Modélisation du couple code source/connaissances (SC/K[193]).

Les cas THALIX et JORAM montrent qu'il y a deux niveaux d'expertise liés au code d'un logiciel libre. (1) Il y a le niveau générique : ce premier niveau[194] que nous nommerons SC/K_1 permet d'exploiter économiquement un logiciel comme un composant pour remplir une fonction dans une architecture logicielle. C'est plutôt le métier de l'intégrateur : celui-ci incorporera une brique open source dont il connaît le fonctionnement général puisqu'il va être capable de créer des connecteurs (plus ou moins fiables en fonction de ses connaissances du code de la solution embarquée) entre la brique et les autres éléments du système.

(2) Il y a le niveau supérieur : le second niveau SC/K_2 permet de maintenir le logiciel. C'est un niveau où il est nécessaire de connaître le code source pour pouvoir créer des extensions, corriger les erreurs du logiciel surtout lorsqu'elles sont d'un niveau élevé et bloquantes au niveau de l'utilisation de l'application. Le premier niveau nécessite un investissement réduit en termes de compétences tandis que le second requiert un investissement lourd pour pouvoir faire évoluer l'application. K_1 et K_2 permettent d'exploiter SC en totalité.

Il convient de noter que pour acquérir (K_1 ; K_2) l'éditeur (E_1) a utilisé les ressources[195] (R_1 ; R_2). Lorsqu'une firme (E_2) souhaite concurrencer l'éditeur elle doit au moins réaliser un investissement de (R_1 ; R_2) pour qu'elle soit en mesure de maîtriser parfaitement la technologie. Toutefois la limite de ce modèle est que la firme (E_2) n'a pas intérêt à utiliser

[192] P_3 = le montant de l'accord cadre passé entre Red Hat et Thales.
[193] SC/K = Source Code / Knowledge.
[194] SC = Code Source ; K_1 = premier niveau de connaissance.
[195] R_1 = correspond à l'investissement nécessaire pour acquérir K_1 et R_2 = l'investissement nécessaire pour acquérir K_2.

autant de ressources car elle pourrait créer un produit concurrent. Au lieu de cela la firme (E_2) souhaitant intégrer la technologie SC entrera en partenariat avec (E_1) pour assurer un niveau de service convenable à ses clients (voir les cas THALIX et JORAM).

Plus généralement, l'industrie du logiciel libre et du logiciel fermé reposent sur deux logiques de création de valeur bien différentes. L'une repose une grande partie de son économie sur le nombre d'utilisateurs puisque ces derniers payent à priori une licence d'utilisation à la firme. La firme fait moins d'argent sur ce qui est autour. Tandis que dans l'open source, les firmes ne valorisent le nombre d'utilisateur que comme un potentiel pour utiliser des services associés. La firme bâtit son modèle économique sur les connaissances acquises en développant le logiciel.

Ce que nos cas prouvent c'est qu'il existe un couplage important entre le code source, les connaissances développées lors de la production du code et par conséquent l'exploitation commerciale d'un logiciel libre. D'autre part, grâce à leur expertise pointue et à la mise à disposition d'une technologie, les éditeurs de logiciels libres peuvent dégager des revenus sur des projets qu'ils n'auraient pas été capables de réaliser soit à cause de leur taille soit à cause de compétences additionnelles dans un domaine.

Plus généralement, le mot *open* n'est pas forcément synonyme d'ouverture parfaite de la concurrence. Le cas OpenXML en est une excellente illustration, ce format est tellement ouvert que personne n'est actuellement capable de l'implémenter… Selon un employé de Novell, le détail des spécifications du format OpenXML tient sur 5946 pages (FIguiere 2007) : il est par conséquent très long d'implémenter ce format.

Dans les deux chapitres précédents [Chapitre 1 et 2], nous avons étudié les différentes stratégies des organisations dans l'open source. Désormais, nous allons nous pencher sur la dynamique stratégique des organisations de l'open source en mobilisant à la fois les dimensions organisationnelles [Partie III], technologiques et économiques [Chapitre 1 et 2].

Chapitre 3. ANALYSE DYNAMIQUE DES REGIMES DE L'OPEN SOURCE.

Dans les précédentes parties, nous avons montré en premier lieu que la communauté d'utilisateurs-développeurs était une organisation issue de la rencontre inédite entre les systèmes de management de la solidarité et les systèmes de production distribués [Partie II].

Ensuite, nous avons vu que ce modèle racine avait connu différentes expansions et qu'il existe aujourd'hui une grande variété d'organisations impliquées dans le développement de logiciels libres. Par conséquent, il était nécessaire de bâtir une typologie contingente prenant en considération les spécificités organisationnelles de ces régimes [Partie III].

Puis, nous avons caractérisé les stratégies d'entreprises dans l'open source en étudiant à la fois les modèles économiques ouverts, hybrides et fermés [Chapitre 1] ainsi que les stratégies industrielles et la concurrence [Chapitre 2].

Grâce à ces différentes analyses, nous sommes désormais en mesure d'étudier la dynamique des régimes de l'open source en intégrant à la fois les dimensions technologiques, économiques et organisationnelles puisque celles-ci sont étroitement liées les unes aux autres dans l'industrie du logiciel [Chapitre 3].

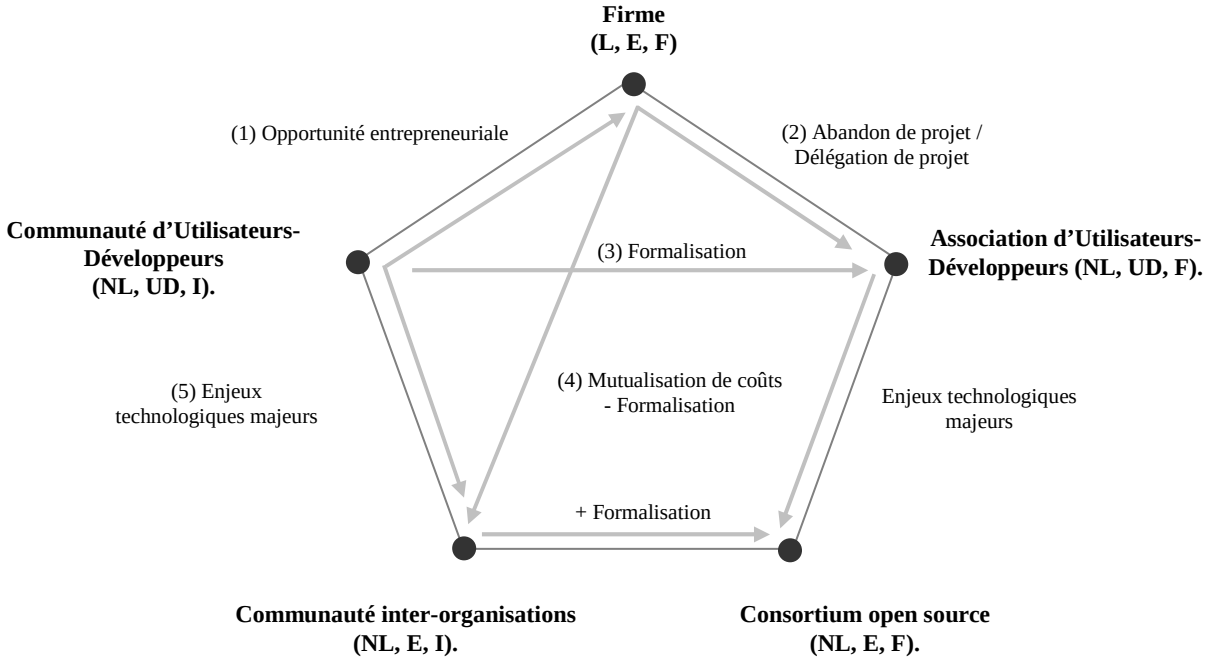
Pour ce faire, nous utiliserons le modèle du pentagone comme grille d'analyse dynamique permettant de suivre l'évolution des régimes de production à travers le temps. Il existe des liens intenses et complexes entre les organisations de l'open source. Ces liens sont d'autant plus complexes et dynamiques qu'ils sont principalement informels. Ceci est à mettre en opposition avec le monde des affaires où chaque prise de participation est plus ou moins connue par le marché. Afin de matérialiser cette dynamique organisationnelle, nous étudierons deux cas sous un angle historique : le cas OpenOffice.org[196] [2.] et le cas Mozilla [3.]. Mais en premier lieu, il convient de revenir sur notre modélisation des régimes de l'open source afin d'expliquer le principe de dynamique des régimes [1.].

1. Modélisation de la dynamique des régimes.

Dans le graphique ci-dessous, nous modélisons le passage d'une forme organisationnelle à une autre en mettant en évidence le(s) principaux déterminant(s) de cette transformation.

[196] Bien que le cas OpenOffice.org a fait l'objet d'une monographie dans la troisième partie, nous analyserons ce cas d'une manière totalement différente.

Figure 27 : Modélisation des transformations organisationnelles.



Le graphique matérialise quelques possibilités de transformations organisationnelles dans l'open source. Le passage d'une communauté d'utilisateurs-développeurs à une entreprise (1) est possible soit de manière simplifiée (NL, UD, I) → (L, E, F). D'après le Directeur des Opérations de Wallix, *« Il y a des communautés qui se transforment en sociétés*[197] *»*. Dalibo est un cas matérialisant la création d'une entreprise par des utilisateurs-développeurs. Nous avons rencontré l'un des fondateurs de Dalibo à qui nous avons présenté nos travaux de manière informelle en mettant en avant le fait que nous étudions la dynamique organisationnelle existant dans l'open source notamment les cas de création de communautés par des entreprises. Notre interlocuteur nous a dit qu'en quelque sorte ils avaient fait le chemin inverse puisqu'ils étaient issus de la communauté PostgreSQL. La communauté PostgreSQL n'est pas à son premier coup d'essai puisque GreatBridge était une autre tentative de société fondée par trois des six *« core developers »* de PostgreSQL (Spiller et Wichmann 2002: 20). D'après le Directeur général de Dalibo l'échec de GreatBridge était dû au fait que PostgreSQL n'était pas encore suffisamment stable pour entrer dans un contexte de production.

Le passage d'une firme à celui de l'association d'utilisateurs-développeurs (2) est aussi possible, soit : (L, E, F) → (NL, UD, F). La plupart du temps, il s'agit d'un acte de délégation

[197] Jean-Noël de Galzain, Directeur des Opérations, Wallix, Entretien avec l'auteur, 6 aout 2007.

du développement d'un logiciel à une fondation ou bien à l'abandon pur et simple d'un logiciel. Mozilla peut-être considéré comme un cas qui matérialise l'échec d'une firme (Netscape) abandonnant un logiciel à une association d'utilisateurs-développeurs (Mozilla Foundation) même si entre les deux, il y a la formation d'une communauté hybride (Mozilla Organization). Pour simplifier, il s'agit de : (L, E, F) → (NL, E, I) → (NL, UD, F).

Le passage de la communauté d'utilisateurs-développeurs à celui de l'association d'utilisateurs (3) est également possible, soit : (NL, UD, I) → (NL, UD, F). Le cas Apache matérialise ce changement. Le développement d'Apache a tout d'abord débuté en 1995 par l'Apache Group, *« une organisation informelle, un groupe de personnes qui se sont accordés pour collaborer sur un travail commun*[198]*. »* Puis, l'Apache Software Fondation (ASF) a été créé il s'agit *« d'une corporation formelle à but non lucratif*[199]*. »*

Le passage de la firme à la communauté inter-organisations (4) est aussi possible, soit : (L, E, F) → (NL, E, I). Le cas OpenOffice.org (OOo) est la preuve vivante de cette transition. Il s'agit de la libération d'un logiciel dont l'enjeu technologique est important. En 1999, Sun libère OOo et créé la communauté OOo, puis en 2003 OOo devient une communauté inter-organisations : (L, E, F) → (NL, E, I).

Le passage de la communauté d'utilisateurs-développeurs au consortium (5) est également possible, soit : (NL, UD, I) → (N, E, F). Le cas du noyau Linux matérialise ce changement.

Bien que les cas OpenOffice.org et Mozilla aient fait l'objet d'un nombre d'études importants, l'angle de lecture dynamique permet de décrire plusieurs phénomènes particuliers qu'il n'est pas possible de déceler avec une lecture statique des cas. Nous présenterons à chaque fois les éléments stratégiques, technologiques et organisationnels particulièrement saillants dans ces deux cas.

2. Le Cas OpenOffice.org : une histoire mouvementée.

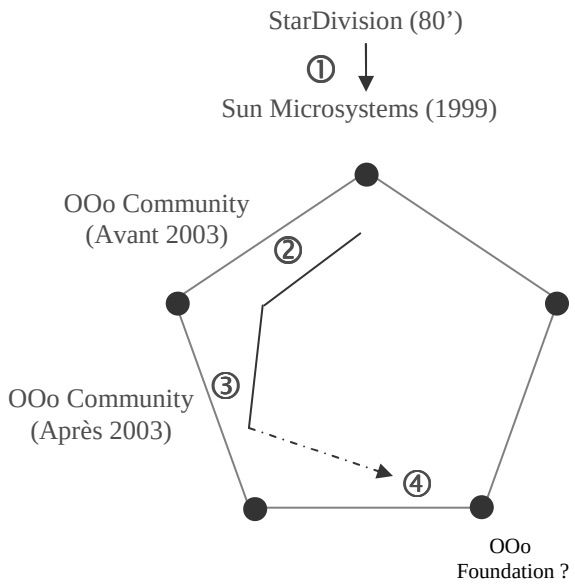
L'histoire du développement d'OpenOffice.org peut-être analysée de trois manières différentes. Nous passerons en revue ces différentes interprétations à l'aide du modèle du pentagone.

2.1. Première interprétation de l'histoire d'OpenOffice.org

[198] Fondateur, Apache Software Foundation, Entretien en ligne avec l'auteur, 28 juin 2006.
[199] Ibid.

L'histoire d'OpenOffice.org (OOo) peut être résumée par le graphique ci-dessous. La description des phases marquantes du développement d'OOo sera réalisée ci-après.

Figure 28 : Schéma de la 1ère interprétation de l'histoire d'OOo.



2.1.1. Phase 1 : le rachat de StarDivision par Sun Microsystems

A l'origine de StarOffice (et par conséquent d'OOo), il y a StarDivision, une société allemande qui lança dans les années 80 une suite bureautique sur un marché de la bureautique très concurrentiel. (Labbé 2003: 7-8). Durant l'été 1999, Sun Microsystems, fit l'acquisition de StarDivision. Par conséquent, Sun devint le propriétaire de tous les produits de StarDivision, notamment son produit phare : StarOffice. En octobre 1999, Sun commercialisa la première version de la suite StarOffice. Très rapidement, Sun sépara le produit StarOffice en deux versions : l'une commerciale et l'autre gratuite.

2.1.2. Phase 2 : la libération de la version gratuite de StarOffice.

Le 19 juin 2000, Sun annonça la libération de la version gratuite de StarOffice (la version StarOffice 5.2.). Finalement, le code source fut publié le 13 octobre 2000. Cet événement marqua le début d'OpenOffice.org (OOo) désignant à la fois le nom de la communauté et de la suite bureautique libre. En créant OOo, Sun chargea la société CollabNet de superviser le développement de la suite bureautique avec le modèle libre. Néanmoins, OOo est une

communauté *« autonome »*, régie par ses propres organes de direction. Il convient de noter qu'à l'époque de la libération d'OOo, le marché de la bureautique était largement dominé par Microsoft Office. Sun a décidé de libérer le code source de la suite bureautique StarOffice qu'elle avait récemment acquise.

OOo fut ouverte sous une double licence : la SISSL et la LGPL. La Sun Industry Standard Source License (SISSL) permettait aux développeurs de conserver secret les modifications apportées au code source d'OOo. La SISSL était à la fois reconnue par la Free Software Foundation et l'Open Source Initiative. La Lesser General Public License (LGPL) permet de lier le code compilé d'un logiciel libre avec d'autres librairies non libres, c'est-à-dire les codes compilés d'autres logiciels.

Les deux parties sont donc distinctes et peuvent faire l'objet de licences différentes. Le code source et la marque OpenOffice.org étaient (et sont toujours) la propriété de Sun. A cette époque, les développeurs souhaitant participer substantiellement au développement d'OOo (c'est-à-dire à plus de 5 lignes de code) devaient signer un contrat nommé *« Copyright Assignment »* qui prévoyait un transfert des droits à Sun Microsystems. Juridiquement parlant cela signifiait que Sun pouvait, en tant que propriétaire du code d'OOo, décider de changer sa licence ou utiliser tout ou partie du code d'OOo dans d'autres produits. Au départ, la présence d'ex-employés de StarDivision était un élément essentiel permettant le bon déroulement du développement distribué puisqu'ils étaient les seuls à pouvoir lire le code d'OOo.

2.1.3. Phase 3 : La mutation d'OOo en communauté inter-organisations.

Le 13 aout 2002, suite à des négociations avec les membres de la communauté, Sun a décidé de changer de type de contrat qu'un développeur devait signer pour participer substantiellement au code d'OOo. C'est ainsi que Sun a créé le JCA (Joint Copyright Assignment) prévoyant un partage de copyright entre l'auteur et Sun. Les deux parties ayant les mêmes droits sur l'œuvre, y compris commerciaux.

L'année 2003 marque l'entrée dans la communauté OOo de Novell et Red Hat. En 2005, c'est au tour de Google et Intel d'entrer dans la communauté. Depuis 2004 environ, OOo et StarOffice partagent le même tronc de développement. Au final, OOo est une communauté principalement composée d'acteurs affiliés à des entreprises.

En septembre 2005, une nouvelle version d'OOo fut lancée : la version 2.0. Cette version fut désormais distribuée sous LGPL uniquement.

En 2007, c'est au tour d'IBM d'entrer dans la communauté, la firme avait déjà utilisé du code OOo pour ses produits (Symphony).

En octobre 2008, la version 3.0. d'OOo succéda à la version 2.0. sous la même licence.

2.1.4. Phase 4 : vers la création d'une fondation OOo ?

A long terme, certains acteurs de la communauté OOo ont formulé le souhait de doter la communauté d'une structure juridique ; à priori, le choix serait celui d'une fondation. Cet élément viendrait renforcer la cohésion du partenariat interentreprises (Segrestin 2004, 2006). La volonté de l'adoption d'une structure juridique du type fondation n'est pas neuve : elle est pratiquement née en même temps que le lancement d'OOo. Cette fondation aurait pour mission : d'une part d'encadrer les opérations, la stratégie technologique, l'incorporation des contributions technologiques ; et d'autre part d'établir des standards ouverts en plus des autres standards de la communauté open source. La fondation OOo serait calquée sur le modèle de l'Apache Software Foundation (ASF), largement accepté dans la communauté open source. Nous expliquons l'influence du modèle de l'ASF sur la communauté OOo par le fait qu'OOo est en partie gérée par CollabNet, une société initiée par un des fondateurs de l'ASF. D'après un membre du Community Council, *« La fondation permettra un rééquilibrage des forces en présence. Les intérêts économiques développés par des sociétés comme IBM ou Sun peuvent être contradictoires et donner lieu à des conflits, c'est d'ailleurs la raison pour laquelle IBM ne souhaite pas participer*[200] *à OpenOffice.org pour le moment, tant que Sun en détiendra le copyright. La fondation, de part la pluralité de ses participants […] permettra un rééquilibrage des forces et un contre poids aux pressions qui pourraient s'exercer*[201]*. »*

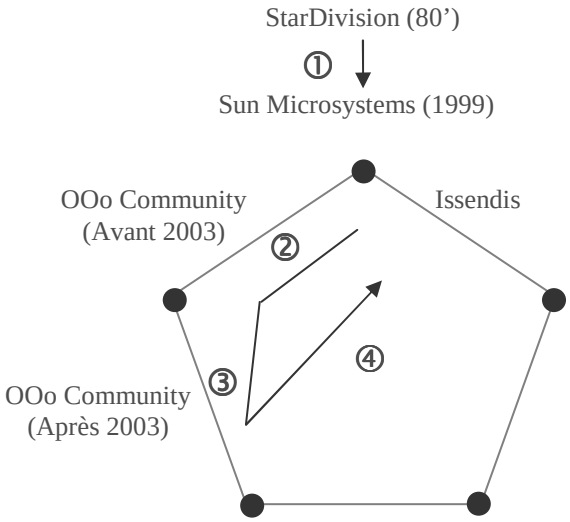
2.2. Autres interprétations de l'histoire d'OOo.

L'histoire d'OOo peut être lue autrement, bien entendu OOo commence toujours de la même manière : le rachat de StarDivision par Sun Microsystems (Phase 1), la libération du code source de la version gratuite de StarOffice (Phase 2), l'entrée d'autres organisations dans la communauté (Phase 3). En revanche, la suite de l'évolution d'OOo peut être interprétée de deux différentes manières.

[200] Au moment de l'entretien IBM ne faisait pas encore partie de la communauté OOo.
[201] Membre du Community Council, OpenOffice.org, entretien en ligne avec l'auteur, le 2 mai 2006.

2.2.1. Phase 4' : La signature d'un accord entre Sun et Issendis.

Figure 29 : Schéma de la 2ème interprétation de l'histoire d'OOo.



Issendis[202] a signé son premier contrat avec Sun en 1999 l'autorisant à distribuer la version gratuite de StarOffice (la version 5.2). Selon Alain Curt, directeur commercial d'Issendis, *« Il y a peu de gens qui à l'époque savait que cette version gratuite, pour la distribuer, il fallait avoir les accords de Sun. Et on a rencontré énormément de partenaires qui distribuaient StarOffice. […] Document que nous avons et que nous avions, alors que beaucoup de société distribuaient, sans savoir quelque part qu'elles violaient la distribution. Il ne s'en souciait pas car le produit était gratuit mais ça ne veut pas dire que l'on peut faire tout et n'importe quoi. Et on a rencontré des petites comme de très grandes sociétés qui ne le savaient pas. Avant de distribuer le produit on a regardé les conditions de licence. Il fallait avoir les accords de Sun Microsystems et ils nous l'avaient donné au préalable[203]. »*

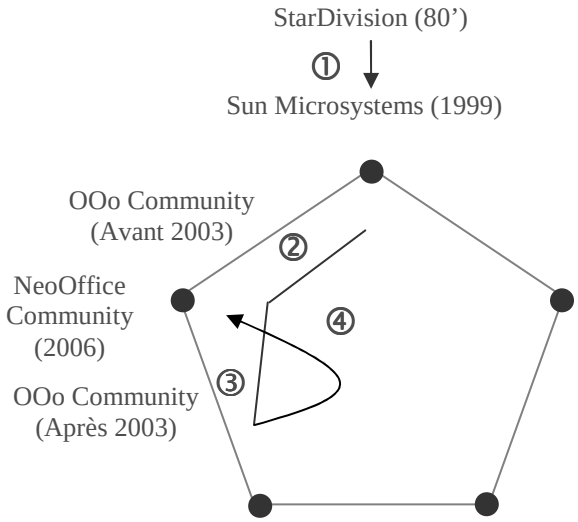
En 2003, la société française Issendis passa un autre contrat avec Sun lui permettant d'utiliser le code d'OOo pour l'intégrer dans un package logiciel. Issendis est une société ayant une activité de conception et de développement de logiciels. La société Issendis a conçu *« Office One »*, un pack de logiciels intégrant un module bureautique basé sur le code d'OOo. Issendis, éditrice d'Office One a utilisé l'opportunité offerte par la LGPL. D'après A. Curt, cette licence permet *« de prendre tout produit qui s'appuie sur la LGPL […] de les modifier en mieux […] et puis vous faire rémunérer pour le travail que vous avez apporté. Mais en aucun cas, on ne s'approprie le travail originel. »* Toutefois, pour que le travail final ne soit

[202] Les informations de cette partie sont basées sur des propos recueillis auprès d'Alain CURT (Directeur commercial - Issendis), le 13 novembre 2006.
[203] Alain Curt (Directeur commercial - Issendis), Entretien téléphonique avec l'auteur, le 13 novembre 2006.

pas soumis à la LGPL, il faut que les deux parties (code originel d'une part, et code fermé d'autre part) soient dissociées.

Issendis ne s'occupe pas de la distribution d'Office One. Issendis a *« plus une vocation de conception, de développement »*, cette société *« n'est pas structurée pour être en relation avec le client final. »*. La distribution d'Office One est assurée par Avanquest, un partenaire d'Issendis. Avanquest dispose d'une organisation lui permettant de s'adresser au client final. En 2006, plus de 190 000 licences d'Office One avaient été vendues et ce en quasi-totalité en France.

2.2.2. Phase 4'' : La création de NeoOffice.



NeoOffice est un *« fork*[204] *»* d'OOo créé pour fonctionner nativement avec le système d'exploitation Mac OSX. NeoOffice est distribué sous GPL. Au moment de la création de NeoOffice, OOo n'était pas nativement compatible avec Mac OS X : il fallait utiliser X11. X11 (ou X Window System) est un système de fenêtrage pour environnement Unix, il permet d'exécuter des applications Unix sur Mac OS X.

NeoOffice a été fondé par deux individus. Le premier était un ancien employé de Sun, il participait au portage d'OOo pour Mac OS X avec X11 et a créé une entreprise de conseil en développement de logiciels. Le second était un contributeur bénévole ayant également travaillé sur le portage Mac OS X avec X11. A l'époque, il avait décidé de participer au portage d'OOo pour Mac OS X en tant qu'utilisateur de Mac OS X.

[204] Un fork consiste à créer *« un nouveau projet en reprenant le code de l'application, mais parallèlement au développement de celle-ci. »*, Développeur Actif 1, Kexi, Entretien en ligne avec l'auteur, 14 mai 2006.

Les fondateurs de NeoOffice travaillaient initialement sur le portage natif d'OOo pour Mac OS X. Des tensions sont nées à partir de la fin 2005, ces dernières ont conduit à une complète séparation des deux groupes durant le courant de l'année 2006. Avant cette scission, deux stratégies coexistaient pour le portage d'OOo sur Mac OS X : NeoOffice/J (Option 1) consistait à *« modéliser de manière interne ce que OOo était en train de faire à ce que OS X attendait via Java »* et NeoOffice/C (Option 2) : consistait à *« faire ce que était en train de faire NeoOffice/J, mais en utilisant Cocoa. »* (Drukenbatman 2005)

La seconde méthode semblait être plus prometteuse en termes de vitesse, d'efficience et d'intégration dans Mac OS X. Du fait du manque de stabilité de la solution utilisant Cocoa, les deux développeurs se sont focalisés sur la solution consistant à utiliser Java.

Les initiateurs de NeoOffice justifient la création puis le maintien de leur fork pour plusieurs raisons. Au départ, la raison motivant la création de NeoOffice concernait la licence d'OOo (SISSL), celle-ci n'était pas appréciée par certains membres de l'équipe chargée du portage d'OOo sur Mac OSX. Désormais, les membres de NeoOffice justifient le maintien d'un logiciel séparé car selon eux cela est plus efficient en termes de ressources financières et en développement. Il y a une autre raison beaucoup plus critique du fonctionnement de la communauté OOo *« OpenOffice.org est financé par Sun Microsystem[s] et son équipe rémunérée est plus concernée par l'accomplissement des objectifs définis par Sun Microsystem[s]. »* (NeoOffice 2009) C'est justement cette raison qui semble la plus crédible puisque d'après un de nos entretiens : *« Il y a peu de développeurs pour ce port [Mac OS X], mais il représente à termes un assez gros marché. Sun n'aidera pas ce port, Apple étant un concurrent. Nous bénéficions maintenant d'un soutien d'Apple Europe parce que nos développeurs ont montré la crédibilité de leur projet dans un port natif pour Mac*[205]*. »*

Apple a contribué au développement d'une version d'OOo pour Mac OS X en offrant du matériel à OOo. La raison justifiant cela est d'après un membre d'OOo, qu'*« il n'y a pas d'alternative crédible à Microsoft Office sur Mac*[206]*. »*

[205] Membre du Community Council 1, OpenOffice.org, Entretien en ligne avec l'auteur, 2 mai 2006.
[206] Ibid.

2.3. Le cas OpenOffice.org ou la réouverture de la concurrence.

Le cas OpenOffice.org (OOo) est riche en enseignements. Tout d'abord, OOo commence par l'histoire d'un logiciel (StarOffice) dont les parts de marché étaient très faibles (pour ne pas dire quasi inexistantes) par rapport à MS Office. Le logiciel StarOffice dégageait peu de revenus pour Sun puisque la firme américaine distribuait déjà deux versions de StarOffice : l'une gratuite, l'autre payante.

L'engagement général de Sun dans l'open source a commencé par l'annonce de la libération d'OOo. Les formats de Microsoft s'étaient largement imposés en tant que standards de fait même si ces derniers (.doc, .xls, .ppt, etc.) n'étaient pas documentés. A des fins d'interopérabilité, les membres communauté OOo ont travaillé sur des formats Microsoft *« ont fait le reverse [engineering] quasi complet du format .xls et publié ce document*[207]*… »* par exemple. L'objectif de Sun Microsystems n'était pas d'améliorer les ventes de StarOffice à court terme, s'il s'agissait de l'objectif : c'est un échec puisque la libération d'OOo *« n'a pas eu d'impact significatif. Ceux qui achetaient StarOffice, on continué à l'acheter et ceux qui ne souhaitaient pas l'acheter ont fait de l'OpenOffice.org*[208]*. »*

L'objectif de Sun Microsystems était plutôt de réouvrir la compétition sur le marché de la bureautique en imposant un standard ouvert dans l'industrie. D'après un membre d'OOo, *« Les acteurs comme : Sun, HP, Novell, IBM : [sont] des sociétés dont l'unique but est de tuer Microsoft afin d'en tirer des profits*[209]*. »*.

Etant donné que la détermination d'un standard ouvert dans la bureautique n'intéresse pas que Sun mais d'autres industriels : progressivement d'autres entreprises se sont jointes à Sun dans OOo créant une véritable communauté inter-entreprise. Bien au-delà d'une simple suite bureautique, le partenariat autour d'OOo, vise à relancer la concurrence sur le marché des systèmes d'exploitation. La bureautique est aujourd'hui un des éléments les plus importants en ce qui concerne l'interopérabilité des systèmes.

Sun utilise d'autres moyens pour définir des standards ouverts dans la bureautique. L'accord avec Issendis prouve bien que la volonté de Sun est de diffuser OOo au plus grand nombre.

[207] Chercheur industriel et expert sécurité des systèmes d'informations, Alcatel-Lucent, Entretien en ligne avec l'auteur, 4 avril 2007.
[208] Technology Advisor, Sun Microsystems, Entretien avec l'auteur, le 15 janvier 2007.
[209] Contributeur, OOo, Entretien en ligne avec l'auteur, 10 mai 2006.

A l'inverse, lorsqu'un projet est contraire aux intérêts de Sun, la firme alloue très peu de ressources. Pour le projet de portage d'OOo sur Mac OS X, il est clair que *« Sun n'aidera pas ce port*[210] *»*. Les propos de l'ex-Lead du projet de portage Mac OS X (jusqu'à fin 2007) sont assez clairs concernant l'état du projet tel qu'il était au moment où il quitta délibérément ses fonctions : *« La version est stable, et tout est contrôlé par Sun. Je m'en suis désintéressé progressivement. [..] Au passage, j'ai bien implémenté les transitions OpenGL (avec Impress) pour la version Mac OS X, mais n'ayant eu ni demande ni remerciement, j'en ai conclu que cela n'intéressait personne, donc je ne m'en occupe plus, et le cws n'est plus maintenu*[211]*. »* C'est la raison principale qui a provoqué aussi la création du *« fork »* NeoOffice.

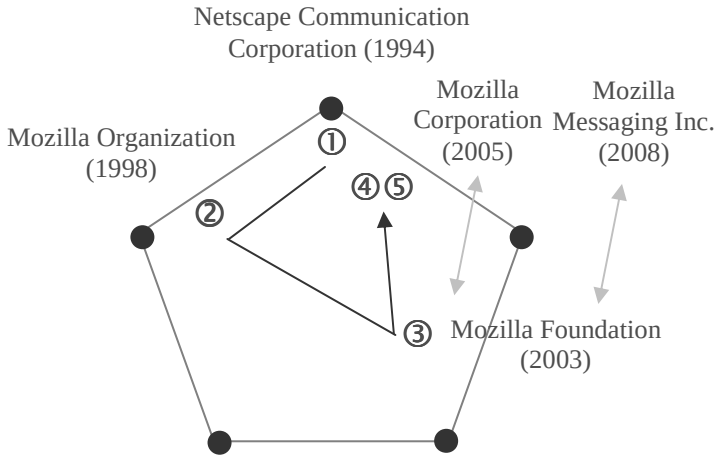
En définitif, le modèle open source peut être considéré d'après le cas OOo comme une réaction de solidarité inter-firmes visant la réouverture de certains marchés. Le système d'exploitation GNU/Linux est aussi un excellent cas matérialisant ce phénomène. Un des membres de Freeworks décrit le développement de Linux de la manière suivante : *« ce sont des communautés autour de systèmes d'exploitation qui sont très puissants… Il y a tout un consortium, une organisation autour de ça*[212]*. »* Depuis sa création, la communauté Linux a beaucoup évolué jusqu'à devenir aujourd'hui une communauté inter-entreprise voire un consortium industriel avec IBM à sa tête.

3. Le cas Mozilla : le marchand au service du non-marchand.

L'histoire de la production de Mozilla peut être divisée en cinq étapes. Le graphique ci-dessous matérialise le passage d'une étape à l'autre en utilisant le modèle du pentagone de manière dynamique. Ce schéma sera tout de suite suivi d'une description de chaque étape.

[210] Membre du Community Council, OOo, Entretien en ligne avec l'auteur, 2 mai 2006.
[211] Ex-Lead, Port Mac OS X, OOo, Entretien en ligne avec l'auteur, 16 juillet 2009.
[212] Développeur-Musicien, Freeworks, Entretien avec l'auteur, 6 juin 2006.

Figure 30 : L'évolution de Mozilla.



3.1. Phase 1 : L'émergence de Navigator.

Netscape Communications Corporation[213] (Netscape) fut fondée en avril 1994 par des anciens du NCSA (National Center for Supercomputing Applications) américain où ils développèrent *Mosaic* : le premier navigateur internet capable d'afficher des images (Wikipedia 2009b). La première version (beta) de *« Navigator »* fut lancée en octobre 1994 et Netscape débuta sa commercialisation en décembre (Jackson 1999: 6). En 1995, Navigator dominait le marché des navigateurs internet avec une part de marché supérieure à tous les autres produits. A partir de ce moment, Navigator fut considéré comme une menace pour la domination de Windows de Microsoft. En effet, Navigator *« fut conçu pour fonctionner dans quelques systèmes d'exploitation et donnait l'opportunité aux développeurs de développer des programmes fonctionnant directement dans le navigateur Netscape. »* (Department of Justice 1998).

Par conséquent, Microsoft mit en place une stratégie afin d'éliminer le navigateur Netscape. Microsoft créa Internet Explorer (IE), son propre navigateur internet. Etant donné que Microsoft savait que Netscape *« réalisait une partie significative de ses revenus grâce à la vente de licences de son navigateur »*, Microsoft décida d'introduire IE dans Windows sans augmenter le prix de celui-ci (Jackson 1999: 68-69). De plus, Microsoft créa tout un lot de partenariats pour distribuer IE sans droits de distribution (Jackson 1999: 70). Microsoft et Netscape commencèrent à distribuer gratuitement leur navigateur respectif de ce fait *« les consommateurs pour la plupart perdirent toute motivation pour payer pour »* un navigateur, (Jackson 1999: 74). En 1996, Microsoft lia technologiquement IE et Windows 95, en d'autres

[213] Pour simplifier nous appellerons cette firme *« Netscape »*.

termes IE partageait des fichiers avec Windows 95, de ce fait la suppression d'IE devenait très difficile sans endommager Windows (Jackson 1999: 80-81).

La politique de Microsoft eut un effet progressif. En janvier 1996, Navigator disposait de 80% des parts d'usage. En novembre 1997, cette part d'usage descendit à 55%. A l'inverse, la part d'usage d'IE bondît de 5% à 36% pour la même période.

3.2. Phase 2 : La libération de Communicator.

Le 31 mars 1998, face à la chute des parts d'usage de Communicator[214], le manque de profits et l'opportunité estimée de l'open source, Netscape libéra le code source de son navigateur et Mozilla était né… Au moment de cette libération, Netscape créa une communauté informelle composée d'employés. Cette structure était nommée la *« Mozilla Organization »* ou Mozilla.org, un groupe d'individus créé pour assurer la coordination du développement de Mozilla. A ce moment là, le code source de Mozilla fut divisé en modules. A la tête de chaque module, il y avait un *« maintainer*[215] *»*.

Dans un premier temps, Mozilla fut libéré sous la Netscape Public License (NPL) qui tentait *« de trouver un compris entre la promotion du développement libre par une firme commerciale et protéger les développeurs de sources libres. »* (Hamerly, Paquin, et Walton 1999). Néanmoins, cette licence fit l'objet de nombreuses critiques par la communauté open source.

Dans un second temps, en réponse à ces critiques, Netscape suggéra que le code originel de Mozilla donné par Netscape et que toutes les modifications apportées à celui-ci étaient couvertes par les termes de la NPL ; et d'un autre côté, les contributions de la communauté seraient couvertes par une nouvelle licence appelée : Mozilla Public License (MozPL). Toutefois, Netscape se réservait le droit : (1) d'utiliser du code NPL dans d'autres produits sans avoir une contamination par la NPL; (2) de réaliser des modifications à la NPL et à la MozPL; (3) de re-licencier du code NPL à des tierces parties. De plus, Netscape conservait le copyright de Mozilla. Le 24 novembre 1998, AOL (à l'époque partenaire de Microsoft) racheta Netscape pour 4,3 milliards de dollars (Jackson 1999: 148).

[214] Après la version 4.0, Navigator a été renommé Communicator.
[215] Un *« maintainer »* est une personne ayant en charge le maintien d'un module ou d'un logiciel.

3.3. Phase 3 : La création de la Mozilla Foundation.

En avril 1999, Frank Hecker, à cette époque employé par Netscape, proposa un article réalisant un bilan un an après la libération de Communicator. Hecker soulignait que Mozilla avait reçu d'importantes contributions de la part de la communauté. Il disait également qu'un problème fut rencontré dans la communauté open source celui-ci était lié au fait que le code de Mozilla ne pouvait pas être intégré dans des applications distribuées sous GPL (par exemple : GNOME[216]). Les licences NPL et MPL étaient incompatibles avec la GPL. D'autre part, Mozilla souffrait d'un problème d'image : il était encore considéré comme étant sous le contrôle d'AOL. F. Hecker proposa de tirer cinq leçons de la libération de Netscape Communicator : (1) les contributeurs à Mozilla ne se limitent pas à ceux qui ont écrit du code ; (2) la qualité des contributions est plus importante que la quantité ; (3) la phase de conception est très importante puisqu'elle permet de résoudre des problèmes avant leur apparition ; (4) la libération d'un logiciel ne suffit pas à une entreprise pour profiter des opportunités offertes par le développement distribué, un travail de fond est nécessaire pour accompagner la démarche, l'open source n'apporte pas de résultats magiques ; (5) la réussite d'un logiciel open source prend du temps (Hecker 1999).

Le 19 juillet 2001, la Mozilla Organisation annonça le début d'une opération de changement de licence vers une triple licence NPL/GPL/LGPL : ce changement concernait plus de 6000 fichiers sous licence NPL. L'objectif de cette démarche visait à régler les problèmes d'incompatibilité de licences susmentionnés.

Le 15 juillet 2003, AOL annonça la dissolution de la Mozilla Organization et la création de la *« Mozilla Foundation »*. La Mozilla Foundation était une corporation à but non lucratif. AOL avait promis de donner deux millions de dollars pour aider à la formation de la Mozilla Foundation. La Mozilla Foundation récupéra toutes les attributions et missions de la Mozilla.org. Elle avait en plus pour mission de promouvoir le développement, la distribution et l'adoption des applications et technologies Mozilla (Mozilla.org 2003). La Mozilla Foundation était une entité totalement autonome d'AOL.

[216] GNOME est un environnement graphique populaire dans les distributions Linux.

3.4. Phase 4 : la création de la Mozilla Corporation.

Le 3 aout 2005, la Mozilla Foundation annonça la création d'une filiale : la *« Mozilla Corporation »*. Dans un premier temps, le nom choisi pour cette nouvelle entité était *« M. F. Technologies »*, puis les membres de la Mozilla Foundation ont décidé d'adopter le nom *« Mozilla Corporation »*. La Mozilla Corporation est *« une « non-tax-exempt organization » dans notre jargon. Autrement dit, c'est une organisation qui est soumise à l'impôt sur les sociétés. C'est important car MoFo [Mozilla Foundation] est exemptée d'impôts, ce qui l'empêche de faire des partenariats avec des sociétés commerciales à une échelle suffisamment grande. Le principal intérêt de MoCo [Mozilla Corporation], c'est qu'elle peut payer des impots, et donc n'est pas soumise aux limites qu'on retrouve chez MoFo*[217]*. »*. La mission de la Mozilla Corporation est de servir les *« buts non lucratifs, l'utilité publique de son [organisation] mère, la Mozilla Foundation, et la vaste communauté Mozilla »* (Mozilla Foundation 2005). Plus spécifiquement, la Mozilla Corporation a pour but de générer des fonds nécessaires au développement de Mozilla à travers des partenariats. D'après Tristan Nitot, il existe deux types de partenariats : technologiques et financiers. *« Les partenariats technologiques sont la plupart du temps « silencieux », car c'est le propre du logiciel Libre : quand Novell ou Red Hat permettent à des ingénieurs de travailler sur le projet Mozilla, il n'y a pas d'annonce particulière, car cela se fait dans le cadre de la collaboration habituelle du logiciel Libre. Il y a bien sûr des exceptions, comme IBM qui fait des communiqués de presse parfois sur l'intégration de technologies développées par ses ingénieurs sur l'accessibilité ou la technologie XForms. Le partenariat avec Adobe est une exception dans la mesure où Adobe a contribué d'un seul coup un gros morceau de code auparavant propriétaire. En complément, il y a des partenariats financiers, avec certains des sites mentionnés dans la zone de recherche (en haut à droite de Firefox)*[218]*. »* C'est justement pour gérer les partenariats financiers que la Mozilla Corporation fut créée. Plus spécifiquement, la Mozilla Corporation a pour but de faciliter la gestion des partenariats Firefox d'un point de vue juridique. Grâce à son partenariat avec Google Inc., la Mozilla Corporation généra plusieurs millions de dollars en 2005.

La Mozilla Corporation permet de rémunérer des contributeurs à Firefox qui jusqu'à 2005 étaient bénévoles. Des transferts de personnels furent réalisés de la Mozilla Foundation à la

[217] Tristan Nitot, Président et Fondateur, Mozilla Europe, Entretien en ligne avec l'auteur, 18 décembre 2006.
[218] Ibid.

Mozilla Corporation. La Mozilla Foundation annonça un bon nombre de partenariats avec des entreprises. Contrairement aux opérations de réorganisation précédentes, la création d'une nouvelle entité n'a pas entraîné la disparition de l'ancienne : cela signifie que la Mozilla Corporation et la Mozilla Fondation existent toutes deux.

3.5. Phase 5 : La création de la Mozilla Messaging Inc.

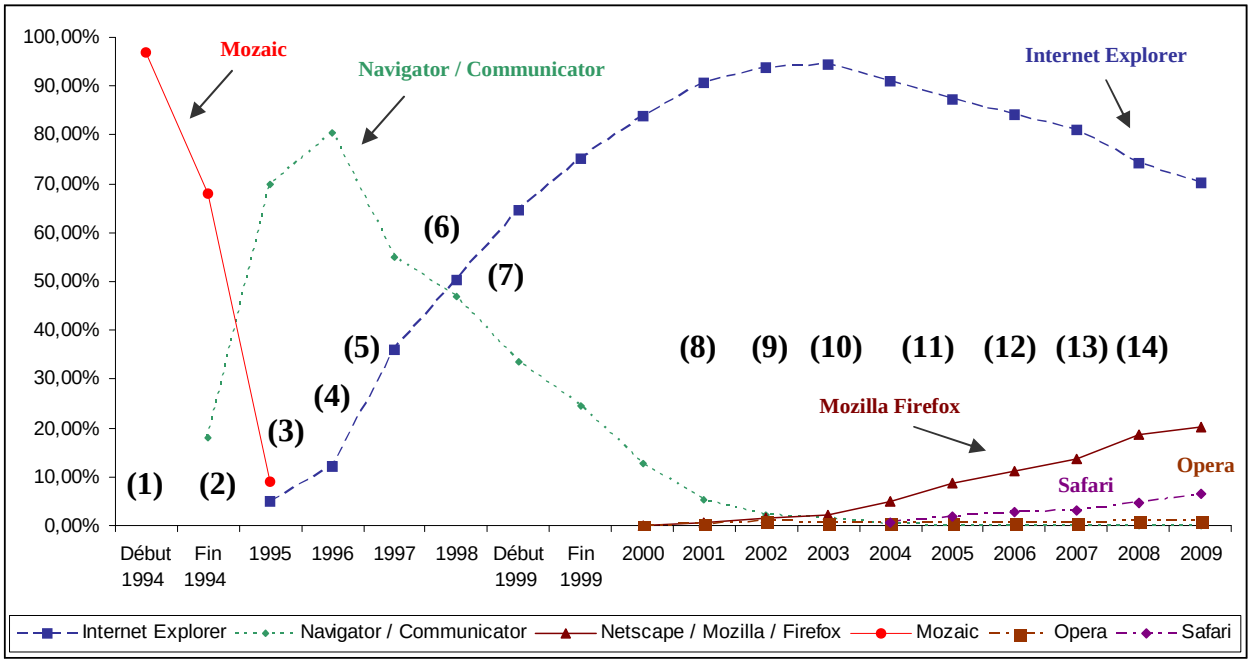
Le 19 février 2008, la Mozilla Foundation annonce la création d'une nouvelle filiale : il s'agit de la Mozilla Messaging Inc. D'après David Ascher, *« c'est une entreprise (une entité soumise à l'impôt), mais avec un unique actionnaire, et c'est une organisation à but non lucratif […]. Par conséquent, nous avons les outils des entreprises à notre disposition, mais notre mission est de remplir les intérêts de notre actionnaire, qui en retour vise à servir l'intérêt public*[219]*. »* Mozilla Messaging Inc. a notamment pour mission de définir des standards en matière de courrier électronique et de calendrier, protéger les données privées des courriels, intégrer la messagerie mobile, la gestion de calendrier et la messagerie instantanée (Ascher, 2008 : 13). Toujours selon D. Ascher, *« A court terme, nous avons un contrat avec la Mozilla Corporation pour assurer le support, maintenir et développer Thunderbird. A long terme, nous n'avons pas encore un modèle économique autonome. Etant donné que notre but est de faire avant l'internet ouvert, c'est acceptable pour notre actionnaire*[220]*. »* Le gros problème posé par le courrier électronique est la multiplication des adresses dans différents services (MySpace, Facebook, Linkedln, etc.), Thunderbird pourrait par exemple fédérer l'ensemble de ces services.

3.6. Mozilla : un paradoxe technologique et économique.

Pour analyser de manière plus approfondie le cas Mozilla, nous proposons le graphique ci-dessous matérialisant les parts d'usage des navigateurs internet depuis 1994 et précise les événements importants pour le développement de Mozilla.

[219] David Ascher, CEO, Mozilla Messaging Inc., Entretien en ligne avec l'auteur, 14 juillet 2009.
[220] Ibid.

Figure 31 : Part d'usage des navigateurs et événements marquants entre 1994 et 2009.



Légende	Evénements	Légende	Evénements
(1)	Avril 1994 : Fondation de Netscape.	**(7)**	Avril 1999 : Problème d'incompatibilité de licences.
(2)	Décembre 1994 : Mise sur le marché de Navigator.	**(8)**	Juillet 2001 : Annonce du changement de licence NPL/GPL/LGPL.
(3)	Juillet 1995 : Mise sur le marché d'IE.	**(9)**	Juillet 2003 : Démantèlement de la Mozilla Organization.
	Intégration dans W95 sans augmentation de prix.		Création de la Mozilla Foundation.
(4)	Aout 1996 : Liaison technologique entre IE et Windows 95.	**(10)**	Adoption du modèle publicitaire.
	Distribution gratuite d'IE et de Navigator.	**(11)**	Aout 2005 : Création de la Mozilla Corporation.
(5)	Signature de divers accords pour promouvoir IE.	**(12)**	Mai 2006 : Processus de re-licensing terminé /
(6)	Mars 1998 : libération de Communicator.		Aout 2006 : Signature d'un accord entre Mozilla Corp., Google et Realnetworks.
	Création de la Mozilla Organization.	**(13)**	Décembre 2007 : Arrêt du développement de Communicator
	Novembre 1998 : Rachat de Netscape par AOL.	**(14)**	Février 2008 : Création de la Mozilla Messaging Inc.

Tableau 21 : Parts d'usage des navigateurs entre 1994 et 2009.

Période	Internet Explorer	Navigator / Communicator	Netscape / Mozilla / Firefox	Mozaic	Opera	Safari	Sources[221]
Début 1994				97%			GVU WWW User Survey*
Fin 1994		18%		68%			GVU WWW User Survey*
1995	5,00%	70%		9%			Department of Justice, 1998 (Navigator) / Jackson, 1999 (IE) **
1996	12,18%	80,45%					GVU WWW User Survey*
1997	36,00%	55,00%					Jackson, 1999**
1998	50,43%	47%					EWS Web Server at UIUC*
Début 1999	64,60%	33,43%					WebSideStory*
Fin 1999	75,31%	24,68%					WebSideStory*
2000	83,95%	12,61%	0,14%		0,14%		TheCounter.com*
2001	90,83%	5,23%	0,71%		0,36%		TheCounter.com*
2002	93,94%	2,31%	1,67%		0,83%		TheCounter.com*
2003	94,43%	1,45%	2,22%		0,66%		TheCounter.com*
2004	90,98%	0,18%	5,10%		0,68%	0,77%	TheCounter.com*
2005	87,25%	0,07%	8,60%		0,71%	1,83%	TheCounter.com*
2006	84,11%	0,05%	11,13%		0,60%	2,80%	TheCounter.com*
2007	81,14%	0,06%	13,81%		0,67%	3,21%	TheCounter.com*
2008	74,24%	0,07%	18,66%		0,89%	4,52%	TheCounter.com*
2009	70,31%	0,06%	20,12%		0,94%	6,44%	TheCounter.com*

* Les données de ces sites internet furent collectées à partir de Wikipedia (2009a).
** Ces données proviennent de Jackson (1999) et Department of Justice (1998).

Il convient tout d'abord de souligner que Mozilla est né à partir de l'échec commercial d'un produit qui pourtant dominait le marché des navigateurs internet. La bataille ayant opposé Microsoft Corporation (Microsoft) et Netscape Communication Corporation (Netscape) n'a pas laissé à Netscape d'autre choix que de libérer le code source de Communicator. Mais ce que montre surtout ce cas, c'est l'échec d'une stratégie ne reposant pas vraiment sur un véritable modèle économique. En fait, nous pensons que Netscape a échoué à construire un modèle économique avec les technologies Mozilla. Le modèle économique basé sur de la prestation de services (modèle dominant dans la valorisation des logiciels libres) n'étaient pas du tout adapté à un navigateur internet. Quels services proposer pour un logiciel dont l'usage est simple (c'est-à-dire sans possibilité de tirer des revenus provenant de prestations de support technique) et ne nécessitant pas d'adaptations particulières pour être utilisé (sans possibilité de proposer des développements spécifiques) ? Ce que le cas Mozilla prouve, c'est que *l'open source commercial doit être accompagné d'une réelle stratégie technologique et un modèle économique adapté.*

[221] Les données de ce tableau proviennent de sources différentes, l'exactitude des données est certainement discutable. Toutefois, ces données nous permettent d'avoir un ordre de grandeur raisonnable pour suivre l'évolution des parts d'usage dans le domaine des navigateurs.

En 1999, un employé de Netscape soulignait que la libération d'un logiciel ne suffisait pas à une entreprise pour profiter des opportunités offertes par le développement distribué : un travail de fond est nécessaire pour accompagner la démarche, l'open source n'apporte pas de résultats magiques (Hecker 1999). Nous défendons l'idée que l'open source n'est pas une stratégie adaptée pour valoriser tous les types de logiciels. De même, le closed source n'est pas une stratégie adaptée pour valoriser tous les types de logiciels. Parfois, conditionner l'utilisation d'un logiciel au paiement d'une licence va à l'encontre des intérêts de certains logiciels. Par exemple, les logiciels dont la valeur repose justement sur le fait qu'ils soient utilisés de manière massive sont peu adaptés à une valorisation de type vente de licences. Les logiciels de Peer 2 Peer (P2P) sont un bon cas illustrant ce phénomène[222].

La mise à disposition du code source ne suffit pas pour développer un modèle d'affaires : il faut une réelle stratégie technologique et économique. D'après le Directeur de la Recherche et de l'Innovation de Thales D3S, *« un modèle qui ne créé pas de valeur est-il durable à long terme ? Non, et on le voit avec Mozilla : il faut trouver un modèle de financement. Certains comme Stallman dirons que oui mais bon*[223]*… »*

La Mozilla Foundation a contourné le problème de la valorisation de Firefox (navigateur internet) en adoptant un modèle économique particulier : le modèle publicitaire. Le modèle économique n'est pas le seul responsable du succès grandissant de Firefox, il y a aussi le fait que Firefox intègre d'une part les dernières avancées en matière technologique mais aussi le fait qu'il soit personnalisable par le biais d'extensions.

En d'autres termes, Mozilla Firefox repose son succès sur deux éléments : d'une part (1) un financement arrivant à conjuguer un logiciel dont la valeur réside dans un usage relativement standard et un modèle économique particulier (le modèle publicitaire) ; d'autre part, (2) un nombre important d'innovations au niveau applicatif (extensions) et du logiciel lui-même (exemple : navigation par onglets, restauration des pages en cas de crash, etc.). Nous pensons que Mozilla Firefox est un logiciel particulièrement atypique : il s'agit d'un cas présentant une *anomalie* au sens de Christensen (2006: 41).

Mozilla Firefox est une anomalie du point de vue de la valorisation car les logiciels libres (orientés *utilisateur-final* ou *end-user*) dont la valeur réside justement dans l'usage adoptent

[222] Dans le chapitre dédié à la caractérisation de l'innovation dans l'open source, nous constatons que l'un des seuls domaines où l'innovation réside dans l'usage dans les communautés d'utilisateurs-développeurs sont des domaines où il est quasiment impossible d'avoir un modèle économique basé la vente de licences. Et qu'intrinsèquement le conditionnement de l'utilisation du logiciel à l'achat d'une licence va à l'encontre de l'utilité procuré par le logiciel. Pour plus en détails sur ce point se référer à la partie portant sur l'évaluation de l'innovation [Partie V].

[223] Serge Druais, Directeur de la Recherche et de l'Innovation, Thales D3S, entretien téléphonique avec l'auteur, 4 aout 2009.

soit le modèle : (1) de la souscription : un modèle consistant à partager le coût de maintien d'un logiciel libre par le biais du paiement d'une souscription annuelle donnant accès aux mises à jour du produit notamment [224] ; (2) du *« Dual-licensing »* : un modèle consistant à vendre une version simple libre et une version élaborée non libre. Ces deux modèles prouvent bien que la valorisation de ce type de logiciel est complexe et que le modèle publicitaire n'est pas adaptable à tous les logiciels. Comme le souligne Tristan Nitot : *« Je ne suis […] pas certain que le modèle économique de Mozilla soit généralisable. »* (Le Monde.fr 2008).

Pour appuyer cette proposition, nous prouverons à travers le logiciel Mozilla Thunderbird (client de messagerie de la Mozilla Foundation) que les dimensions économiques et technologiques sont étroitement liées et que le modèle de financement de Mozilla Firefox n'est pas adaptable à tous les logiciels.

Thunderbird est un logiciel orienté utilisateur-final dont la valeur réside principalement dans l'usage. Traditionnellement, ce type de logiciel est valorisé par le biais de la vente de licences. Il y a peu de valeur autour du logiciel notamment en termes de services associés même si l'installation de certains logiciels peut nécessiter de la configuration spécifique avec un serveur de messagerie.

Du point de vue technologique, la stratégie de *« Thunderbird est simple : de rattraper le retard accumulé, et construire une plateforme de développement qui pourra permettre des fonctions uniques. […] Depuis plusieurs années, les clients email (Outlook, Notes, Thunderbird, etc.) n'ont pas vraiment innové. Thunderbird, étant basé sur la même base que Firefox, devrait pouvoir évoluer beaucoup plus vite, grâce a une architecture qui permet des innovations décentralisées -- n'importe qui peut construire un « add-on ». On verra*[225] *! »*

Du point de vue économique, le modèle publicitaire n'est pas adaptable à Thunderbird. En tous cas, à ce stade de développement étant donné le nombre réduit d'utilisateurs de Thunderbird : 2,4 % des utilisateurs en entreprises et 3 % auprès du grand public, voir les données ci-dessous.

[224] Dans un certain sens le modèle de la souscription et la vente de licence sont très similaires. Dans le cas de la souscription, la licence consiste à faire payer l'utilisateur au fur et à mesure tandis qu'avec la licence classique il paye le produit une fois mais n'a pas accès aux nouvelles versions du logiciel.
[225] David Ascher, CEO, Mozilla Messaging Inc., Entretien en ligne avec l'auteur, 15 juillet 2009.

Tableau 22 : Parts d'usage des clients de messagerie.

Entreprises*

Clients	Parts d'usage
Outlook	36%
Hotmail	33%
Yahoo! Mail	14%
Gmail	6%
Apple Mail	4%
Windows Live Mail (Desktop)	3%
Thunderbird	2,40%
iPhone	1,30%
Lotus Notes	0,20%
AOL Mail	0,10%
Total	100%

Particuliers*

Clients	Parts d'usage
Yahoo! Mail	29%
Outlook	27%
Hotmail	25%
Apple Mail	4%
Gmail	4%
Comcast	3%
AOL Mail	3%
Thunderbird	2%
Windows Live Mail (Desktop)	2%
iPhone	1%
Total	100%

*Source : FingerPrint Septembre (2008)

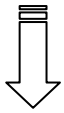
D'après le CEO de Mozilla Messaging Inc., *« En gros, oui -- il serait possible d'insérer des pubs dans Thunderbird (comme Gmail le fait, ou Eudora l'a fait), mais ça n'est pas un projet qu'on considère à ce stade*[226]*. »*

Toujours selon David Ascher, la structure organisationnelle de Mozilla permet d'être plus efficient en termes de ressources car la stratégie de Mozilla Messaging Inc. est la suivante : *« D'abord, on s'applique à faire un client email qui est utilisé par autant de personnes que possible. Plus le nombre d'utilisateurs est grand, moins on a besoin de générer de [dollar par] personne pour faire en sorte que Thunderbird puisse être maintenu à long-terme. Vu qu'on est une société qui n'a pas un but purement lucratif, il est envisageable d'avoir un Thunderbird qui ne fait que payer ses frais […] c'est un avantage énorme par rapport à la plupart des entreprises commerciales qui doivent générer toujours plus de profits chaque année. […] On n'a pas les mêmes besoins de profitabilité, et nos coûts sont moins élevés que ceux d'autres parce que beaucoup du travail est fait par des volontaires (par exemple, toutes les traductions, beaucoup de QA, etc.)*[227]*. »*

[226] Ibid.
[227] Ibid.

Question de Recherche 3 :

QR3 : Comment les logiciels libres sont-ils valorisés ?

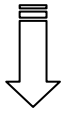


Thèse 3 :

TH 3 : **Les logiciels libres sont valorisés par des modèles : (1) ouverts, (2) hybrides et (3) fermés (basés sur des stratégies de verrouillage). Toutefois, les logiciels orientés utilisateur final sont difficilement valorisables avec un modèle ouvert. Seules trois possibilités sont empiriquement identifiées : le financement publicitaire, le modèle de la souscription ou la cession de licence (modèle hybride ou fermé).**

Question de Recherche 4 :

QR 4 : Quel est l'impact de l'open source sur l'industrie du logiciel ?



Thèse 4 :

TH 4 : **Les logiciels libres sont utilisés comme des composants génériques sur la base desquels la valeur est créée dans l'industrie permettant l'indépendance technologique. Néanmoins leur pleine exploitation économique nécessite deux catégories de connaissances : l'une est générique (intégration), l'autre repose sur une expertise (édition).**

Dans la partie précédente [Partie IV], nous nous sommes intéressés aux stratégies d'entreprises dans l'open source. En commençant tout d'abord par nous pencher sur la question de la valorisation économique des logiciels libres [Chapitre 1]. La littérature faisait état de deux types de modèles économiques : (1) les stratégies ouvertes reposant essentiellement sur de la prestation de services et (2) les stratégies hybrides consistant à coupler à la fois des revenus provenant de licences et de l'autre des services associés. Grâce à nos investigations exploratoires, nous avons identifié (3) une catégorie de modèles économiques basée sur ce que nous nommons des stratégies de verrouillage. Ces stratégies introduisent un véritable paradoxe en faisant un bien privé à partir de ressources collectives [Thèse 3] inversant en quelque sorte la logique du *« private-collective model »* (Stuermer et al. 2009; von Hippel et von Krogh 2003; von Krogh 2006). Certaines de ces stratégies sont de pures violations de licences libres tandis que d'autres tirent parti de nouvelles technologies et tentent de bâtir des valorisations directes du logiciel libre.

Ensuite, nous nous sommes tournés vers l'étude des stratégies industrielles et la concurrence [Chapitre 2]. En premier lieu, nous avons exposé comment Thales, un groupe industriel spécialisé dans l'électronique embarqué et la création de systèmes d'informations dans des domaines hautement sensibles, a adopté une stratégie d'intégration de composants open source. La stratégie de Thales souligne que les briques libres sont essentielles pour l'indépendance technologique et économique du groupe et de ses clients mais ce n'est pas sur la partie libre que la création de valeur est réalisée : ce sont des composants génériques réalisant des opérations maitrisées incorporées dans un contexte industriel rigoureux [Thèse 4]. Thales combine des technologies libres et non libres pour créer des solutions sur-mesure. Toutefois, Thales ne libère pas de technologie sur lesquelles la firme construit sa légitimité. Pour renforcer le caractère générique et industriel des composants libres : l'architecture est garantie par le systémier, les composants pouvant varier.

Puis, nous nous sommes également intéressés à la dynamique concurrentielle dans l'open source à travers le prisme de la théorie CK (Hatchuel et al. 2002; Le Masson et al. 2006) proposant une séparation entre d'un côté les concepts et de l'autre les connaissances. Ce cadre fut mobilisé pour étudier deux cas : le premier était Thalix, un système de gestion du trafic aérien conçu par Thales à partir de composants libres et le second est Joram, un système de messagerie basé sur une plateforme à agents distribuée issue d'une collaboration industrielle et académique édité par ScalAgent Distributed Technologies. A partir de ces deux cas, nous dérivons une modélisation du couple connaissance (K) et code source (SC) en découvrant que le couplage SC/K fait apparaître deux niveaux d'expertise pour tirer profit d'un logiciel libre :

(1) le niveau *générique* permettant d'utiliser le logiciel comme un composant dans un système d'information et (2) le niveau *édition* permettant d'assurer le maintien, la création d'extensions et la réalisation de modifications profondes du logiciel [Thèse 4].

Enfin, nous nous sommes intéressés à la dynamique stratégique de deux cas : OpenOffice.org et Mozilla, étudiés d'un point de vue historique [Chapitre 3]. Le cas OpenOffice.org décrit comment l'open source pouvait être utilisé par un ensemble d'entreprises pour relancer certains marchés caractérisés par une domination monopolistique. En l'occurrence les marchés de la bureautique et des systèmes d'exploitation. Le cas Mozilla présente de son côté un cas très particulier d'utilisation du marché pour financer le non-marchand. En outre, ce cas suggère également que la valorisation des logiciels orientés utilisateurs-finaux est difficile avec un modèle libre. Pour ce type de logiciels seules trois possibilités sont identifiées : le financement publicitaire, le modèle de la souscription ou la cession de licence [Thèse 3].

Après avoir présenté les stratégies d'entreprises dans le domaine de l'open source, nous nous intéresserons à la question de l'innovation dans les régimes libres objet de la prochaine partie [Partie V].

PARTIE V. L'INNOVATION DANS LES REGIMES DE L'OPEN SOURCE : COMPENSER LES DEFAILLANCES DE L'OFFRE MARCHANDE.

Dans cette partie, nous étudierons l'innovation dans les régimes de l'open source et dans une moindre mesure dans les régimes closed source. Pour ce faire, nous réaliserons d'abord un état de l'art sur la *« user innovation »* et l'innovation dans l'open source [Chapitre 1]. Puis, nous présenterons les apports de la web-based Delphi : une technique grâce à laquelle nous traiterons la question de l'innovation du point de vue du concept général du logiciel. Plus spécifiquement, nous étudierons les types d'innovations produites dans les communautés d'utilisateurs-développeurs [Chapitre 2]. Pour finir, nous aborderons l'innovation au niveau applicatif et nous nous pencherons sur les spécificités respectives des modèles open et closed source [Chapitre 3].

Chapitre 1. L'ETAT DE L'ART SUR LA *« USER INNOVATION »* ET L'EVALUATION DE L'INNOVATION DANS L'OPEN SOURCE.

L'évaluation des innovations conçues par les utilisateurs et les communautés open source ont fait l'objet de quelques études empiriques. Ces études sont très contradictoires quant à leurs résultats. Nous vous proposons tout d'abord de suivre la construction de la littérature sur les communautés indépendantes d'utilisateurs-développeurs produisant des logiciels libres [1.]. Puis, nous discuterons des limites des travaux portant spécifiquement sur l'évaluation de l'innovation des logiciels libres [2.].

1. De la *« user innovation »* aux communautés open source.

Dans cette section, nous présenterons les travaux identifiant le caractère innovant des utilisateurs [1.1.]. Puis, nous rendrons compte d'une récente étude posant la question de la radicalité des innovations utilisateurs [1.2.]. Et enfin, nous traiterons des conditions justifiant l'émergence de communautés d'utilisateurs-développeurs indépendantes [1.3].

1.1. L'identification des utilisateurs comme source d'innovations.

Eric von Hippel (1988) a été l'un des premiers auteurs à défendre l'idée que les *« innovations produits »* n'étaient pas seulement développées par les fabricants originels de ces derniers. En s'intéressant à l'origine de 253 innovations dans 9 industries (instruments scientifiques, semi-conducteurs, etc.), von Hippel démontra empiriquement que les sources des innovations pouvaient être multiples[228] (von Hippel 1988: 3). Ainsi, il a identifié trois principales sources d'innovation : les fabricants les fournisseurs et les utilisateurs. von Hippel montra qu'il y avait une variation des sources d'innovations en fonction des industries étudiées. Il expliquait cette variation par le fait que les innovations sont développées par ceux qui en attendent le plus grand bénéfice.

Le principal apport des recherches menées par von Hippel est l'identification d'innovations créées par des utilisateurs ou ce qu'il nomme les *« user innovations »*. Grâce à cette étude, von Hippel introduisit le concept de *« lead user »* permettant de prédire la source d'une innovation (von Hippel 1988: 102-16). La technique du lead-user consiste à identifier les utilisateurs ayant deux caractéristiques particulières : (1) ils ont des besoins qui deviendront généraux sur le marché des mois ou des années en avance ; (2) ils sont positionnés de telle manière qu'ils profitent de manière significative en obtenant une solution à leurs besoins (von Hippel 1988: 107). Lorsque les lead-users sont identifiés tout ce qu'il reste à faire est de les aider à générer des innovations. Dans des recherches plus récentes, von Hippel propose de transférer les activités de conception aux utilisateurs via des « *toolkits* » (Franke et von Hippel 2003a: 1199; von Hippel et Katz 2002: 821).

1.2. Le rôle des utilisateurs dans la définition d'une offre radicalement innovante.

Récemment, Scheid (2009) a soutenu une thèse portant sur l'implication des utilisateurs dans un processus de conception d'une offre logicielle sur-mesure mais dans le domaine du logiciel fermé. Scheid a étudié ce phénomène par le biais du suivi longitudinal de plusieurs projets d'une firme spécialisée dans le *« text mining »*, un domaine permettant d'extraire du

[228] Il convient de souligner ici que Robert Allen avait déjà mis en évidence la variété des sources de l'innovation en identifiant quatre catégories d'innovateurs : (1) les centres de recherche à but non lucratif et agences gouvernementales ; (2) les firmes par le biais des activités de R&D ; (3) les inventeurs individuels ; (4) la collective-invention (Allen 1983: 1).

contenu textuel et de réaliser différentes opérations de traitement par le biais d'outils sémantiques (Scheid 2007: 156).

La firme propose une gamme de produits que Scheid considère comme innovante pour deux raisons : cette offre résulte de la combinaison d'algorithmes d'analyse sémantique et statistique et elle introduit une rupture dans les usages (Scheid 2007: 156-57).

Scheid souligne également que le suivi longitudinal de deux projets lui a permis de *« voir émerger une offre innovante unique : le couplage d'un logiciel d'extraction (ESoft) et de catégorisation de l'information (Ksoft) avec un logiciel de gestion de base de connaissance (ASL), destiné au secteur du publishing. »* (Scheid 2007: 160). Scheid considère cette combinaison de solutions comme une innovation radicale. Pour notre part, nous estimons que le couple *« radical/incrémental »* introduit une forte subjectivité du fait de la prédéfinition de ce qui est considéré comme radical et incrémental. Comme Garcia et Calantone l'ont montré, il y a de multiples terminologies et ces terminologies font l'objet de débat quant à leur définition (Garcia et Calantone 2002). Il est aujourd'hui nécessaire d'avoir une approche de l'innovation limitant la hiérarchisation arbitraire des types d'innovations car celle-ci conduit à une réelle orientation des résultats des recherches. Plus spécifiquement, dans l'industrie informatique, nous pensons qu'il est nécessaire d'avoir une typologie de l'innovation contingente permettant de catégoriser l'innovation à la fois *ex-post* et surtout *ex-ante* : c'est-à-dire une typologie capable d'identifier le type d'innovation au moment de la définition du concept du logiciel et non une fois que celui-ci est vendu (logiciel fermé) ou plus généralement mis à disposition des utilisateurs (logiciel libre).

Dans le cas développé par Scheid, les utilisateurs ne codent pas (Scheid 2007: 167) mais ils sont réellement acteurs du processus de définition de l'offre par le biais de l'expression de leurs besoins en n'hésitant pas à faire appel à plusieurs firmes (Scheid 2007: 165). Toutefois l'auteur souligne qu'il n'est pas vraiment possible de parler de communauté d'utilisateurs car les utilisateurs ne disposent pas des connaissances nécessaires leur permettant de modifier le logiciel (Scheid 2007: 166).

1.3. Les conditions de l'émergence de communautés d'utilisateurs indépendantes.

Jusqu'ici les travaux de von Hippel portaient sur des utilisateurs de produits existants offerts sur un marché. Mais la démocratisation d'internet a donné naissance à de nouveaux

régimes de production produisant des *« biens informationnels »* (Benkler 2002: 13; Kogut et Metiu 2001a: 249). Les recherches de von Hippel ont constitué un cadre de plus en plus pertinent pour expliquer le phénomène des communautés d'utilisateurs produisant pour eux-mêmes ou ce que von Hippel nomme des *« réseaux d'innovation horizontaux par et pour des utilisateurs »* (von Hippel 2002).

Selon von Hippel trois conditions facilitent les communautés d'utilisateurs. (1) les utilisateurs doivent avoir suffisamment de motivations pour innover ; (2) les utilisateurs doivent partager leurs innovations ; (3) les innovations créées doivent pouvoir faire concurrence à leurs alternatives commerciales (von Hippel 2001: 84). Les communautés d'utilisateurs-développeurs impliquées dans le développement de logiciels libres rassemblent ces trois conditions. Nous vous proposons de montrer comment ces trois conditions sont bel et bien remplies.

Condition 1 : la motivation pour innover. Avec le logiciel libre, le bénéfice attendu des modifications est supérieur au coût de ces améliorations. Cette situation est notamment due au fait que les logiciels libres ont une structure modulaire permettant que des modifications soient réalisées avec un impact réduit sur la structure du logiciel (MacCormack et al. 2006: 1015-16). Cette modularité assure en grande partie la coordination entre les développeurs (Benkeltoum 2006: 58; Scheid 2007: 153). La *modularisation* est une pratique bien connue et bien établie dans l'industrie informatique. D'après Scheid, cela correspond à *« la décomposition d'un système complexe en sous-systèmes quasi autonomes, pouvant être conçus de manière indépendante »* (Scheid 2007: 153).

D'après Kogut et Metiu, cette pratique a été mise en évidence par Cusumano (1991) qui montra que *« la conception de logiciel est passé de l'art à des tâches routinières manipulant des modules standardisés »* (Kogut et Metiu 2000: 5). Selon von Krogh et al., la littérature existante sur le développement de logiciels fermés suggère que diviser le code source en modules a trois effets bénéfiques sur un projet logiciel : cela favorise la transparence du projet, réduit les barrières pour contribuer et rend possible la spécialisation grâce à une utilisation plus efficiente des connaissances (von Krogh et al. 2003a: 1218).

En comparant la modularité de cinq couples de logiciels libres et fermés, MacCormack et al. ont prouvé que pour quatre des cinq couples, les logiciels libres avaient une structure plus modulaire que les logiciels fermés (MacCormack, Rusnak, et Baldwin 2007: 15).

Condition 2 : le partage des innovations. L'ouverture du code des logiciels libres est considérée comme une incitation au partage de l'innovation (AlMarzouq, Zheng, Rong, et

Grover 2005: 775). Pour Henkel, la General Public Licence (GPL) est un outil permettant aux utilisateurs de partager leurs innovations (Henkel 2003b: 22).

Condition 3 : le caractère concurrentiel des innovations. Du fait que le logiciel a la nature d'un *« bien informationnel »* (Benkler 2002; Kogut et Metiu 2001a: 249) cela implique plusieurs choses. Tout d'abord, cela signifie qu'il s'agit d'un bien non rival (Maggioni 2002: 6; Rullani 2007: 3) : l'utilisation ne provoque pas la destruction ou la dégradation du bien. Ensuite cela signifie que le logiciel peut être distribué à un coût nul (Letellier 2008: 2) ou quasiment nul si l'on prend en compte les coûts d'infrastructure.

2. Les limites de la littérature existante sur l'évaluation de l'innovation des logiciels libres.

Dans cette seconde section, nous discuterons des limites des travaux portant spécifiquement sur l'évaluation de l'innovation des logiciels libres [2.]. Premièrement, nous présenterons une limite importante de la littérature sur l'open source : la confusion entre processus innovant et produit innovant [2.1.]. Deuxièmement, nous analyserons les limites de quelques études proposant d'évaluer l'innovation des logiciels libres [2.2.]. Troisièmement, nous mettrons en évidence le manque de critères pertinents pour évaluer l'innovation dans l'industrie du logiciel [2.3]. Et quatrièmement, nous rendrons compte de la vision controversée des acteurs du terrain interrogés concernant les logiciels libres [2.4.].

2.1. La confusion entre processus innovant et produit innovant.

Dans la littérature, la plupart des chercheurs considèrent que les produits open source sont innovants car leur processus de conception est innovant. Selon Klincewicz, beaucoup d'auteurs emploient de manière synonyme les termes *« open source development »* et *« open source innovation »*, le risque est d'utiliser le mot *« innovation »* comme par effet de mode (Klincewicz 2005: 3).

Ainsi, pour Rossi le logiciel open source représente en lui même une disruptive innovation et peut être considéré comme un exemple d'open innovation (Rossi 2009: 154). Pour d'autres, le modèle de développement open source est une *« radical innovation »* (Bonaccorsi et al. 2006: 1086). Kogut et Metiu (2000) utilisent l'expression *« E-Innovation »* pour décrire une nouvelle technique de développement où les participants sont des individus

géographiquement dispersés. von Hippel considère les communautés open source comme des *« réseaux d'innovation par et pour des utilisateurs »* (von Hippel 2002). von Hippel et von Krogh décrivent ce qu'ils nomment le *« Private-Collective Model »* en faisant référence à un modèle d'innovation où les participants utilisent leurs propres ressources (privées) pour contribuer à la constitution d'un bien collectif (von Hippel et von Krogh 2003: 213).

Plus récemment ce modèle a été étendu aux communautés hybrides mettant en relation une firme (Nokia), des partenaires et des développeurs bénévoles (Stuermer et al. 2009). Le problème avec ce modèle est qu'il n'y a pas de distinction entre ce qui est innovant, ce qui l'est moins et ce qui n'est pas innovant. Par exemple, Nokia propose à des entreprises concurrentes d'utiliser la plate-forme qu'elle a conçue sur d'autres mobiles. Les auteurs parlent de diffusion de l'innovation ou *« spreading the innovation »* (Stuermer et al. 2009: 180) alors que le phénomène mis en évidence n'est pas le partage d'innovation mais d'un socle technologique générique sur lequel Nokia ne créé pas de valeur. Nokia préfère garder le contrôle sur la couche haute des applications et elle ne partage pas à ce niveau (Stuermer et al. 2009: 178). La raison est que c'est justement sur cette partie haute qu'elle créé de la valeur. Ce cas est typiquement similaire à la stratégie des intégrateurs comme Thales dans le domaine du middleware. Ce que Nokia partage c'est une sorte de *« middleware »*. Pour preuve, un membre de Nokia interrogé par les auteurs souligne que la base créée par Nokia fut réutilisée par un concurrent à Nokia. Toutefois il ajoute aussi le fait que ce que *« notre truc n'était pas si révolutionnaire »* (Stuermer et al. 2009: 182). En d'autres termes, on se demande pourquoi la notion d'innovation est employée pour décrire des portions de code génériques[229] n'ayant pas grand-chose d'innovant. La notion de *« private-collective innovation »* a donc une sérieuse limite dans la mesure où elle considère tout comme une innovation et cela au même niveau.

De nombreux auteurs dans la littérature sur l'open source considèrent toute modification de logiciel comme une innovation. Ainsi, Hicks et Pachamanova nomment le retour des modifications réalisées par des utilisateurs sur le code d'un logiciel libre comme un phénomène de propagation d'innovation utilisateurs (Hicks et Pachamanova 2007: 318).

En étudiant le développement du noyau Linux, Lee et Cole ont montré que les développeurs avaient deux fonctions : une fonction d'assurance qualité et une fonction innovation. Pour l'innovation, les développeurs suggèrent de nouvelles fonctions ou écrivent des correctifs (Lee et Cole 2003: 637).

[229] Les auteurs parlent eux-mêmes de *« generic frameworks »* (Stuermer, Spaeth et von Krogh, 2009 : 185).

Franke et von Hippel ont montré que dans les domaines où les besoins sont très hétérogènes, il est plus pertinent de donner la possibilité aux utilisateurs de satisfaire leurs propres besoins grâce à un *« innovation toolkit »* plutôt que d'essayer de trouver une solution aux besoins moyens avec quelques produits (Franke et von Hippel 2003b).

De même, Osterloh et Rota soulignent que du fait que le logiciel est un produit informationnel, les activités de production et d'innovation sont pratiquement confondues (Osterloh et Rota 2007: 159). Selon Dahlander and Wallin, les logiciels libres sont un cas extrême d'open innovation (Dahlander et Wallin 2006: 1243). Deek et Mchugh défendent aussi la thèse que le paradigme de l'open source favorise l'innovation (Deek et McHugh 2007: 7).

2.2. L'analyse des études proposant d'évaluer l'innovation produit des logiciels libres.

2.2.1. Les évaluations basées sur des études de cas.

Tuomi a été l'un des premiers à poser la question de la validité de l'argument défendant l'idée que l'open source est plus innovant que le closed source. Selon Tuomi, cet argument est intéressant mais il dépend largement de ce *« que l'on entend par innovation »*. Si l'on se place du point de vue de l'innovation produit, Linux doit davantage être considéré *« comme un projet d'ingénierie et d'implémentation »* que comme une réelle innovation. En effet, l'architecture sur laquelle est basée Linux est Unix, elle était innovante au début des années 70 (Ilkka Tuomi 2005: 436). Nous voyons bien que l'évaluation réalisée par Tuomi est très sommaire et basée sur le cas Linux donc difficilement généralisable à l'ensemble des logiciels libres.

Plus récemment, l'évaluation de l'innovation des logiciels libres en tant que produit fut étudiée par Deek et Mchugh en ces termes: *« est-ce que l'open source et innovant ou imitatif »* ? En réponse à cette question, Deek et Mchugh disent *« la réponse est un peu des deux »* (Deek et McHugh 2007: 7). D'un côté, ils expliquent qu'il y a des logiciels libres étant seulement des imitations de logiciels fermés existants. D'un autre côté, ils expliquent que le logiciel libre est la source de la plupart des innovations rendant la création d'internet possible. Toutefois, bien que l'argumentation de Deek et Mchugh semble juste, ils ne montrent pas comment faire la distinction entre ce qui est innovant et ce qui ne l'est pas. De plus, Deek et Mchugh ne font pas une distinction claire entre les logiciels initiés par des utilisateurs-

développeurs (exemple : Linux Kernel, BitTorrent), ceux qui furent libérés[230] (exemple : Mozilla, OpenOffice.org) et ceux issus de la recherche fondamentale (exemple : Apache Web Server, protocole TCP/IP).

Les deux études précédentes sont peu approfondies en ce qui concerne la technique d'évaluation. Nous vous proposons de nous intéresser aux deux études proposant une évaluation quantitative de l'innovation des logiciels libres.

2.2.2. Les études quantitatives proposant une évaluation de l'innovation des logiciels libres.

A. L'étude de Klincewicz (2005).

Klincewicz a réalisé une étude portant sur les 500 logiciels les plus actifs de SourceForge reposant uniquement sur sa propre évaluation. Plus particulièrement, Klincewicz *« se focalise sur l'importance des modifications pour le produit entier, perçu par les individus développant le projet. »* (Klincewicz 2005: 9). La première limite de cette étude est que la base d'évaluation repose sur la description faite par les concepteurs, de ce fait, les concepteurs sont en quelque sorte *juge et partie.* En d'autres termes, nous nous demandons comment peut-on analyser l'innovation d'un produit uniquement à partir d'informations fournies par les concepteurs de celui-ci ?

Klincewicz base son auto-évaluation sur les travaux de Dahlin et Behrens qui (2005) en ce qui concerne l'identification des innovations radicales. Dahlin et Behrens soutiennent l'idée qu'une technologie radicalement innovante est réussie si elle est : *« (1) nouvelle ; (2) unique et a un impact sur les technologies futures »* (Dahlin et Behrens 2005: 717). Selon Klincewicz, le critère de nouveauté et le caractère unique est satisfait si aucune fonctionnalité comparable n'existait avant la date du lancement du produit. Klincewicz ajouta la dimension de plate-forme car certains logiciels sont développés spécifiquement pour une plate-forme (Klincewicz 2005: 7). Klincewicz utilisa la grille reproduite ci-dessous pour évaluer les innovations (Klincewicz 2005: 8).

[230] Cela signifie un logiciel fermé qui a été mis en open source par son propriétaire.

Tableau 23 : Grille d'évaluation de l'innovation (Klincewicz, 2005).

	Nouvelle technologie	**Nouveau pour une plateforme**	**Technologie existante**
Nouveau marché	Radical invention (rupture)		Innovation marketing
Marché existant	Modification de technologie	Modification de plateforme.	Pas d'innovation

La grille d'évaluation proposée par Klincewicz présente d'importantes limites. (1) La première concerne la notion de *« marché »*. Il n'y a pas vraiment de marché pour le logiciel libre puisque la plupart des logiciels libres sont distribués gratuitement par leur concepteur et il n'y a que peu de logiciels libres vendus en tant que tel. La plupart du temps, ce qui est vendu est plutôt des services associés au logiciel.

(2) La seconde limite concerne ce qui est considéré comme une invention radicale. Klincewicz considère comme une innovation de rupture l'application d'une nouvelle technologie pour une nouvelle plate-forme en créant un nouveau marché. La question qu'il convient de soulever ici c'est qu'est-ce qu'une nouvelle technologie ? Comment la reconnaît-on ? Dahlin et Behrens (2005) proposent deux critères, cette technologie doit être nouvelle et unique. Le problème avec cette définition, c'est d'identifier la base de comparaison permettant de certifier qu'elle est nouvelle et unique. La notion d'invention sous-entend une forte intégration de nouvelles connaissances. Qu'en est-il des produits basés sur des technologies anciennes mais provenant d'autres domaines tout en répondant à un marché existant ? Ou ce que Hargadon et Sutton nomment le *« knowledge-brokering »* ou portage de connaissances. Hargadon et Sutton ont montré que les innovations ne provenaient pas forcément de la création d'une nouvelle technologie mais aussi du portage de connaissances d'un domaine où celle-ci est évidente à un autre domaine où elle ne l'est pas du tout (Hargadon et Sutton 2000; 2003). Dans notre étude, nous développerons le cas de Mute qui est une bonne illustration du phénomène du portage de connaissances.

En appliquant le diptyque nouvelle technologie et nouveau marché, on considèrerait également le bateau à vapeur comme non innovant. En effet, le bateau à vapeur inventé par Jouffroy d'Abbans est un produit qui répondait à un marché existant : le transport naval et c'est un produit qui utilisait une technologie existante déjà largement employée dans les mines depuis 75 ans donc une technologie existante. Pourtant le bateau à vapeur est bien l'une des plus grandes inventions du XIX^{ème} siècle (De Hoefer 1858).

L'étude de Klincewicz montre que 87,2% des logiciels libres étudiés sont non innovants, 10,6% des modifications de plate-forme, 1% des innovations radicales, 0,8% des modifications de technologie et 0,6% des innovations marketing. Ce que l'étude de Klincewicz prouve c'est qu'il est facile d'identifier les logiciels non innovants et les modifications de plateformes mais qu'il est complexe d'identifier les autres catégories de logiciels notamment les innovations radicales. Nous pensons que c'est justement à cause du fait que la grille d'analyse des logiciels n'a pas été empiriquement testée au préalable, ni bâtie sur des critères validés par des experts du domaine.

B. L'étude de Rossi (2009).

Une seconde étude empirique fut menée (Lorenzi et Rossi 2008) puis (Rossi 2009). Rossi propose une étude comparative entre les solutions basées sur des logiciels libres et les solutions *« propriétaires »* (expression de l'auteur). Plusieurs limites importantes sont identifiées dans cette étude.

(1) Premièrement, Rossi prouve que les indicateurs classiques de l'innovation (brevets, marques, etc.) ne sont pas applicables à toutes les industries et plus particulièrement à celle du logiciel. Rossi propose donc de choisir des critères spécifiques pour mesurer l'innovation dans le domaine du logiciel. Les critères d'évaluation sont classés en trois dimensions.

(a) La première dimension consiste à mesurer l'innovation pour la firme. Elle repose sur un indicateur unique consistant à mesurer si le logiciel est nouveau pour la firme (indicateur 1). Mais la question qu'il convient de se poser ici, c'est en quoi rapporter une innovation au niveau de la firme renseigne sur le niveau d'innovation d'un logiciel ? Lorsqu'une firme lance un produit c'est qu'elle ne l'a pas lancé avant… Il y a une circularité certaine dans ce critère qui ne nous semble pas pertinent.

(b) La seconde dimension comprend deux indicateurs et vise à mesurer si le logiciel est innovant pour le marché. Dans le premier critère, on demande si le logiciel est plus innovant dans le sens où celui-ci satisfait mieux les besoins et les demandes des utilisateurs (indicateur 2). D'un côté, on est dans une métrique favorisant d'office les logiciels libres puisqu'ils sont orientés vers les besoins de l'utilisateur (voir notre étude plus loin). De l'autre côté, cela ne prend en compte qu'une dimension de l'innovation à savoir la logique qui va de l'utilisateur à la firme. Les travaux de Christensen ont démontré qu'il y avait une catégorie d'innovations qui ne répondaient pas au marché mais en créait un nouveau en redéfinissant les conditions de

la concurrence. C'est ce que Christensen nomme les *« disruptive innovations »*. Cette métrique ne prend donc pas compte cette catégorie de produits (Christensen 1997, 2006).

Le second critère de cette dimension mesure le degré de nouveauté du logiciel d'un point vue technologique (Indicateur 3). Comme pour l'étude de Klincewicz, il y a une focalisation sur la notion de nouvelle technologie comme s'il s'agissait de la seule source d'innovation alors que les travaux de Hargadon et Sutton ont montré que d'anciennes technologies pouvaient être à l'origine d'innovations importantes. De même, Christensen (2006) met en évidence le fait que toutes les *« disruptive innovations »* ne sont pas basées sur des technologies radicales et toutes les technologies radicales ne sont pas des *« disruptive innovations »* (cf. notre réflexion sur les critères permettant de mesurer l'innovation).

(c) La troisième dimension proposée par Rossi est la plus générale, elle contient deux critères. Le premier interroge sur le fait que le logiciel contient un module difficile de trouver ailleurs (Indicateur 4). Nous nous demandons comment un expert peut connaître la structuration d'un logiciel qu'il n'a pas utilisé et dont il n'a à priori pas étudié le code source. Les experts de cette étude avaient à leur disposition une brève description du produit et de la firme (Rossi 2009: 159). Nous nous demandons vraiment en quoi cette métrique est pertinente. De plus, même si on peut estimer que le logiciel libre permet de vérifier l'innovation des modules car l'accès au code source est normalement possible. Comment mesurer l'innovation des modules d'un logiciel non libre ? Si l'on n'a pas accès au code source et que celui-ci n'a pas été utilisé…

Le second critère porte sur la nouveauté de la plate-forme logicielle (Indicateur 5) (Rossi 2009: 160). Les deux indicateurs (Indicateurs 4 et 5) sont basés sur la notion *« new to the world »* (Indicateur 4 : *« modules new to the world »* et Indicateur 5 : *« platform new to the world »*). D'après, Christensen la notion de *« new to the world »* est ambiguë, il souligne que *« lorsqu'une innovation ne peut pas être décrite relativement à un produit ou une technologie préexistants, on peut dire alors qu'elle était nouvelle pour le monde. »* (Christensen 2006: 48).

L'ensemble des indicateurs sont *« market-oriented »* ou *« need-oriented »*. Qu'en est-il des logiciels ne répondant pas à un besoin mais le créent ? Les *« disruptive innovations »* dont Christensen parle et les logiciels créant un nouvel usage ? (cf. l'étude empirique). Nous pensons qu'avec ces critères, il y a une proportion anormale de logiciels considérés comme innovants. Pour preuve, il y a plus d'un tiers des produits (35%) considéré comme *« nouveau pour le marché »* en général un tiers des produits ont reçu un haut score d'innovation ce qui est largement inhabituel dans une industrie (Rossi 2009: 162).

(2) Deuxièmement, Rossi souligne le fait que sur chacune des dimensions, les logiciels libres étudiés sont plus innovants que les logiciels non libres (Rossi 2009: 163) et que le modèle de développement des logiciels open source est efficace (Rossi 2009: 165). Plusieurs limites peuvent être mises en évidence.

Tout d'abord, concernant le modèle de production open source. Nos investigations sur le terrain ont montré qu'il n'y avait pas un modèle d'organisation dans l'open source mais une grande variété d'organisations résumées dans le modèle du pentagone [Partie III]. Si l'on se place du point de vue de la division du travail et du génie logiciel, il n'y a pas, d'après les experts de l'industrie que nous avons interrogé, fondamentalement de différences entre le développement de logiciels libres et non libres ni en matière de structuration des équipes, ni en matière de génie logiciel. Fuggetta soulignait que *« les informaticiens (et en particulier les ingénieurs logiciel) n'ont souvent pas considéré l'approche open source comme une réelle rupture dans le développement de logiciels : ils ont ignoré ou bien négligés cela. »* (Fuggetta 2003: 77). Ces éléments sont par conséquent en contradiction avec ce que suggèrent certains auteurs (Rossi 2009: 154).

Ensuite, nous nous posons la question de qu'est-ce qui est réellement étudié ? L'une des questions de recherche de l'auteur est la suivante : *« are programs based on FOSS solutions more innovative than proprietary ones »* (Rossi 2009: 153). Nous voyons bien qu'ici ce n'est pas le logiciel libre qui est évalué mais la solution qui se base sur du logiciel libre. C'est une nuance très importante. D'autre part, l'auteur a souligné lui-même que 167 entreprises sur les 323 de l'échantillon proposaient *« des logiciels basés sur des logiciels libres à leur clients, souvent en les mélangeant avec des offres propriétaires »* (Rossi 2009: 157). Dans quelle mesure l'auteur a-t-il considéré les produits de ces firmes ? Sont-ils libres ou fermés ? Il semble que la séparation entre les logiciels libres et non libres n'est pas évidente et de ce fait la base de comparaison est très discutable. En effet, dans une autre étude l'auteur a montré que les firmes avaient majoritairement des modèles économiques hybrides couplant logiciels libres et fermés (Bonaccorsi et al. 2006).

(3) Troisièmement, l'ensemble de l'analyse repose sur le jugement de trois experts ce qui nous paraît être bien peu si l'on observe le nombre de logiciels évalués (134). En premier lieu, il faut se demander comment un expert peut-il évaluer un logiciel qu'il ne connaît pas ? Peut-il évaluer un logiciel sur la seule base de la description du logiciel qui est faite sur le site internet de l'éditeur ? Peut-il avoir un jugement cohérent et fiable en se basant sur cela ? Lors de notre étude exploratoire nous nous sommes rapidement rendu compte qu'il y a une saturation des experts à partir de 30 logiciels (cf. notre étude). Nous nous demandons

comment un expert peut-il avoir un jugement cohérent pour 134 logiciels. Alors que l'évaluation de 20/30 logiciels populaires nous a semblé être un grand maximum par rapport à la charge que représentait l'évaluation. De même quel est l'intérêt de calculer des moyennes lorsqu'il n'y a que trois évaluations ?

2.3. Le manque de critères pertinents pour évaluer l'innovation dans l'industrie du logiciel.

Dans les précédentes études proposant d'évaluer l'innovation des logiciels libres, il y a plusieurs limites : la plus importante est sans doute le manque de critères pertinents pour mesurer l'innovation. De même, l'ensemble des métriques choisis ne sont uniquement capables d'évaluer l'innovation *ex-post* et il n'est pas possible de dire à l'avance ou *ex-ante* quel logiciel est innovant ou pas et quel type d'innovation celui-ci propose. Pourtant des travaux mirent en évidence les limites d'une telle approche (Danneels 2004: 251; Le Masson et al. 2006: 84). Il est de plus difficile d'accepter ces critères car ils présentent d'importantes limites. L'évaluation proposée par Deek et McHugh est basée sur des cas sélectionnés. Ces auteurs ne suggèrent pas de critères permettant de distinguer entre un logiciel open source innovant et imitatif (Deek et McHugh 2007). Dans les deux études quantitatives (Klincewicz 2005; Rossi 2009), l'ensemble des indicateurs utilisés n'ont pas été testés empiriquement et reposent sur des typologies génériques dont les limites furent démontrées (Garcia et Calantone 2002).

De manière plus générale, il y a deux grands paradigmes dans la littérature sur l'évaluation de l'innovation : le paradigme *« incremental/radical »* (mobilisé dans les études de Klincewicz (2005) et Rossi (2009)) et le paradigme *« disruptive/sustaining »*.

Dewar et Dutton font la distinction entre les *« radical innovations »* et les *« incremental innovations »*. Les radical innovations contiennent un haut niveau de nouvelles connaissances alors que les innovations incrémentales contiennent un bas niveau de nouvelles connaissances (Dewar et Dutton 1986: 1422). La notion de *radical product innovation* se réfère à des produits basés sur des nouvelles technologies (Chandy et Tellis 1998: 476; Danneels 2004: 252; Govindarajan et Kopalle 2006: 13). La plupart du temps, les *radical product innovations* remplacent les anciens produits. Chandy et Tellis donnent l'exemple de produits remplacés par des innovations radicales : la machine à écrire par le traitement de texte ; le télégraphe par le téléphone ; l'appareil photo à plaque de verre par l'appareil à pellicule celluloïd. Mais ce

n'est pas toujours le cas: le micro-onde n'a pas remplacé les autres fours (Chandy et Tellis 1998: 477).

Christensen propose de faire la distinction entre deux types d'innovations: les *« sustaining innovation[s] »* et les *« disruptive innovation[s] »* (Christensen 1997: 15). D'un côté, les *« sustaining innovations »* sont des innovations correspondant à une amélioration des produits existants selon les valeurs du marché dominant. D'un autre côté, les *« disruptive innovations »* introduisent une proposition de valeur n'étant pas attendue par le marché (Christensen 1997: 10-11; Christensen et Overdorf 2000: 71-72). Ce sont ces types d'innovations qui ne sont pas prises en compte dans les études de Klincewicz (2005) et Rossi (2009).

A court terme, les *« disruptive innovations »* ont une performance inférieure aux produits existants si ceux-ci sont analysés sur la base des valeurs du marché dominant. Les *« disruptive innovations »* sont une sorte de pari à long terme : la concurrence n'est pas basée sur des critères préétablis mais plutôt sur la définition de nouvelles règles pour la concurrence (Christensen 1997: 9). Plus récemment, Christensen a fait la distinction entre deux types de *« disruptive innovations »* : les *« low-end disruptions »* et « *new-market disruptions »* (Christensen et Raynor 2003: 51; Christensen 2006: 48). Le concept de *« disruptive innovation »* a récemment fait l'objet d'un profond débat dans la littérature sur le développement de produits. Le point de départ de ce débat est un article de Danneels (Di Benedetto 2006: 1) dans lequel il prétend que Christensen ne donne pas de critères précis permettant de déterminer lorsqu'une technologie est disruptive ou pas (Danneels 2004: 247). L'article de Danneels provoqua un haut niveau d'émulation auprès des universitaires de différentes disciplines (Danneels 2006; Govindarajan et Kopalle 2006; Henderson 2006; Markides 2006; Slater et Mohr 2006; Tellis 2006).

Les notions de *« radicalness »* et de *« disruptiveness »* ne sont pas comparables car elles ne mesurent pas la même chose. La notion de *« radicalness »* est basée sur des aspects technologiques alors que la notion de *« disruptiveness »* met l'accent sur des aspects de marché (Govindarajan et Kopalle 2006: 13-14). Les *« disruptive innovations »* ne sont pas forcément des innovations radicales et les radical innovations ne sont pas forcément disruptive (Govindarajan et Kopalle 2006: 14; Slater et Mohr 2006: 26). La matrice ci-dessous propose la mise en évidence de cette distinction.

Tableau 24 : Couplage des paradigmes *« incremental/radical »* et *« sustaining/disruptive »*.

		Dimension *« Incremental/Radical »*	
		« Incremental »	***« Radical »***
Dimension *« Sustaining/Disruptive »*	***« Sustaining »***	*« Sustaining I »* (basé sur une technologie incrémentale)	*« Sustaining II »* (basé sur une technologie radicale)
	« Disruptive »	*« Low-end Disruptive Innovation »*	*« High-end Disruptive Innovation »*

Comme Garcia et Calantone l'ont montré, l'abondance des terminologies et le débat autour de leur définitions (Garcia et Calantone 2002) rend difficile d'évaluer l'innovation en utilisant ces notions. Aujourd'hui, la littérature manque d'une théorie générale de l'innovation applicable à toutes les industries. De plus, dans notre recherche il n'est pas possible de prendre en considération la performance de marché comme dans les travaux de Christensen (Christensen 1997; Christensen et Raynor 2003; 2006) étant donné que notre recherche est centrée sur des logiciels libres créés et maintenus par des communautés d'utilisateurs-développeurs. Ces logiciels ne sont pas vendus mais distribués gratuitement. De ce fait, nous avons choisi de bâtir un cadre analytique contingent afin de décrire les innovations de l'industrie du logiciel libre. Nous le présenterons dans la prochaine section.

2.4. Le logiciel libre vu par quelques praticiens.

La dernière remarque qu'il convient de faire concerne l'opposition entre la vision des universitaires et des praticiens vis-à-vis des logiciels libres. Rossi soutient l'idée que d'après les études de Tuomi (2005) et Klincewicz (2005) les logiciels libres sont peu innovants ; d'un autre côté, au contraire les praticiens considèrent que les logiciels libres ont un haut niveau de qualité et de fiabilité selon un article de The Economist du 16 mai 2006[231]. Sur le terrain, lorsque l'on demande à des acteurs de l'open source si les logiciels libres sont innovants nous obtenons des réponses en relative contradiction avec la vision citée par Rossi (2009).

« Les logiciels open source sont au mieux un assemblage de fortune pour dépanner. J'ai besoin de trucs qui [..] marchent. Je n'ai pas besoin de passer des semaines à essayer de faire

[231] A noter que nous n'avons pas retrouvé l'article intitulé *« Open-source business »* du 16 mai 2006 cité dans l'article de Rossi comme source défendant l'idée que les praticiens considèrent le logiciel libre comme innovant. Cet article n'a pas été publié le 16 mai mais le 16 mars 2006. De plus, dans cet article il n'y a pas une telle idée. Nous trouvons même un passage prouvant au contraire le manque de clarté sur la manière dont l'open source est innovant. Le verbatim de ce passage est le suivant : *« However, it is unclear how innovative and sustainable open source can utimately be. »* (The Economist 2006: 65).

des trucs avec [un logiciel] open source que je peux faire en un jour avec un truc propriétaire. […] Vous devez blaguer. Rien ne me vient à l'esprit des 15 ans que j'utilise des logiciels open source. Sérieusement, la plupart des trucs vont et viennent[232]*. »* (Ingénieur, Fournisseur de services IT).

« Mon avis sur les logiciels libres est qu'ils copient en général l'existant, du moins pour ceux que j'utilise le plus, et que je m'en sers plutôt comme boîte à outils[233]*. »* (Développeur., Google Inc.).

« Je pense que la plupart des applications open source ne sont pas terriblement innovantes, et même s'il y a des exceptions, il a tellement de différents champs/catégories de logiciels, et à l'intérieur desquelles différentes applications sont innovante de différentes manières – par conséquent il est difficile de vraiment les comparer et les classer. A la fin mon choix semble arbitraire. Ce n'est pas toujours [simple] de séparer les "logiciels que j'aime" et "les logiciels vraiment utiles" des "logiciels qui sont innovants" (en effet quelque chose peut être innovant sans avoir une réelle utilité du tout, par exemple : les jeux) ; […] Je pense que quelque chose peut être innovant, si les idées dans ce logiciel ont vraiment été largement copiés[234]*. »* (Directeur général, TshwaneDJe Human Language Technology).

« La principale innovation des logiciels libres est toutefois la liberté du logiciel, et je pense que le concept le plus innovant provenant de la communauté du logiciel libre est la GPL et le concept de copyleft[235]*. »* (Consultant, Freelance).

Face à la vision proposée ces acteurs du terrain et aux nombreuses contradictions existantes dans la littérature, nous avons décidé de mener une recherche afin d'apporter des réponses concrètes à ce débat. Nous pensons que rechercher à savoir si tous les logiciels libres sont innovants ou pas n'a pas beaucoup de sens. En effet, il existe différents régimes de production dans l'open source : un logiciel libre n'est pas forcément développé par une communauté géographiquement dispersée : certains logiciels sont maintenus par une seule entreprise. Un logiciel n'est jamais que le fruit d'un processus de conception. Il convient donc d'étudier les différentes configurations une à une. Pour cela il est nécessaire d'adopter une méthodologie alternative.

[232] Expert 81, Ingénieur, Fournisseur de services IT, Entretien en ligne avec l'auteur.
[233] Expert 7, Développeur, Google Inc., Entretien en ligne avec l'auteur.
[234] Expert 104, Directeur général, TshwaneDJe Human Language Technology, Entretien en ligne avec l'auteur.
[235] Expert 120, Consultant en logiciels libres et étudiant en cinéma, Freelance, Entretien en ligne avec l'auteur.

Chapitre 2. DE LA METHODE DELPHI A LA CONSTRUCTION DE LA *« WEB-BASED DELPHI »* : ETUDE EMPIRIQUE DE L'INNOVATION.

Dans le précédent chapitre, nous avons vu que la littérature défend la thèse que les communautés d'utilisateurs sont une source considérable d'innovations et que les communautés open source constituent un exemple typique de ce type de réseaux (von Hippel 2002). De plus, la littérature présente plusieurs limites dont les deux plus importantes sont : d'une part, la confusion entre processus de conception innovant et produit innovant ; et d'autre part, le manque de critères pertinents pour évaluer l'innovation *ex-post* et *ex-ante*.

Les communautés open source sont considérées comme des réseaux d'innovation d'utilisateurs (von Hippel 2002). La question qu'il convient donc de poser est la suivante : quels types d'innovations les utilisateurs-développeurs conçoivent-ils ? Pour répondre à cette question, nous avons décidé de focaliser exclusivement notre attention sur l'évaluation de logiciels libres créés totalement par des communautés d'utilisateurs-développeurs. Par conséquent, tous les logiciels analysés dans ce chapitre furent créés par des utilisateurs-développeurs : cela signifie sans aucune intervention des firmes pour le concept original (logiciel fermé libéré) ou pour le financement partiel (logiciel libre sponsorisé). En d'autres termes, nous étudierons des logiciels libres initiés par ce que West et O'Mahony appellent « *community founded* » (West et O'Mahony 2005).

Cette partie de notre travail propose de répondre aux questions ci-dessous. (1) Comment évaluer l'innovation d'un logiciel libre ? (2) Est-ce que le résultat d'un processus de conception innovant est toujours innovant ? (3) Quels types d'innovations les communautés d'utilisateurs-développeurs du logiciel libre produisent ?

Pour répondre à ces questions, nous présenterons le problème de mesure de l'innovation rencontré et notre méthode consistant à coupler des méthodes qualitatives et quantitatives pour le dépasser [1]. Puis nous exposerons le principe général de l'approche *« web-based Delphi »* : une méthodologie contingente et innovante pour étudier l'innovation dans l'industrie du logiciel [2.]. Ensuite, nous détaillerons cette méthode issue du couplage entre différentes techniques de mesure de l'innovation [3.]. Pour finir, nous discuterons des implications de notre recherche.

1. D'un problème de mesure au couplage de méthodes qualitatives et quantitatives pour étudier l'innovation.

1.1. Le problème de mesure de l'innovation.

Le problème rencontré lors de notre démarche était de savoir comment évaluer l'innovation d'un logiciel. D'une part, la littérature portant sur l'évaluation de l'innovation ne contient pas de théorie générale applicable à toutes les industries. D'autre part, les recherches précédentes portant sur l'évaluation de l'innovation dans le domaine du logiciel libre présentaient trois limites importantes : (1) la confusion entre processus de conception innovant et produit innovant (Dahlander et Wallin 2006: 1243; Deek et McHugh 2007: 7; Hicks et Pachamanova 2007; Kogut et Metiu 2000: 318; Osterloh et Rota 2007: 159; von Hippel 2002; von Hippel et von Krogh 2003: 213) ; (2) de sérieux défauts dans la technique d'évaluation (Deek et McHugh 2007: 7; Klincewicz 2005; Lorenzi et Rossi 2008; Rossi 2009); et (3) un manque de critères pertinents pour mesurer l'innovation (Rossi 2009). De ce fait, nous avons recherché à construire nos propres critères pour mesurer l'innovation.

1.2. Les techniques d'évaluation de l'innovation.

Il y a trois façons de mesurer l'innovation : (1) le *« self-assessment »* (Rowe et Wright 1999: 371) ou l'auto-évaluation : une méthode consistant à choisir ce qui est innovant et ce qui ne l'est pas de manière arbitraire ; (2) l'évaluation par un collège d'experts ou la technique Delphi (Peter, Ferrell, et Stewart 1999) ; (3) l'évaluation par les utilisateurs : il s'agit de demander à des utilisateurs de noter les produits qu'ils utilisent. Nous vous proposons de nous pencher sur les avantages et limites de chacune des techniques d'évaluation existantes adaptées aux logiciels libres.

Tableau 25 : Avantages et limites des techniques d'évaluation adaptées aux logiciels libres.

Technique	Avantages	Limites
Auto-évaluation ou *« self-assessment »*	- Maîtrise des paramètres d'évaluation. - Nombre des évaluations non limités.	- Manque de légitimité des critères d'évaluation. - Conditionnement des résultats en fonction des critères sélectionnés arbitrairement.
Delphi technique ou *« expert-assessment »*	- Légitimité de l'évaluation. - Réduction du biais d'évaluation.	- Non connaissance des critères d'évaluation. - Aucune maîtrise des paramètres d'évaluation. - Connaissance du logiciel indispensable. - Nombre limité des évaluations.
Evaluation par les utilisateurs ou *« user-assessment »*	- Connaissance pointue des logiciels utilisés. - Légitimité de l'évaluation.	- Non connaissance des critères d'évaluation. - Manque d'objectivité du fait que certains utilisateurs sont à la fois juge et parti (concepteurs et évaluateurs).

Nous voyons bien que chacune de ces techniques a des avantages mais aussi des limites. Au vu de ces dernières, nous avons d'emblée écarté la technique basée sur l'évaluation par les utilisateurs car l'objet de notre recherche est d'étudier l'innovation des logiciels conçus par des utilisateurs-développeurs. Nous avons également considéré que l'auto-évaluation était telle qu'elle insatisfaisante et ce par manque de critères d'évaluation validés empiriquement. C'est une des limites que nous avons relevé dans les précédentes recherches portant sur l'évaluation de l'innovation des logiciels libres (Deek et McHugh 2007; Klincewicz 2005; Lorenzi et Rossi 2008; Rossi 2009). La seule technique qui nous semblait la plus adaptée est l'évaluation par un collège d'experts ou ce que Rowe et Wright nomment la technique Delphi (Rowe et Wright 1999).

2. De la technique Delphi à une *« web-based Delphi »* : une vue générale.

Nous avons choisi de suivre et d'adapter la suggestion de Danneels proposant d'utiliser la technique Delphi (Rowe et Wright 1999) pour *« intégrer des estimations des trajectoires technologiques et de marché »* (Danneels 2004: 251). Selon Rowe et Wright, la technique Delphi fut développée dans les années 40 par une équipe de RAND Corp. lorsqu'ils étaient impliqués dans un projet de l'US Air Force. Cette technique peut être utilisée lorsqu'il y a un manque de données adéquates et lorsque le jugement humain est nécessaire (Rowe et Wright 1999: 354). Pour Rowe et Wright, il y a quatre conditions pour qu'une technique puisse être nommée « *Delphi » :* l'anonymat, l'itération, le feedback contrôlé et l'agrégation statistique des réponses.

Notre recherche fut menée par le biais d'internet avec des experts répartis dans le monde (26 pays sont représentés) c'est la raison pour laquelle nous n'avons pas adopté l'*idéal* décrit par Rowe et Wright. Nous avons plutôt appliqué une variante de la technique Delphi telle qu'elle a été décrite par les mêmes auteurs (Rowe et Wright 1999: 354-55).

Nous avons choisi d'utiliser l'évaluation d'experts mais nous avons été confrontés à un problème quantitatif. En effet, pour avoir des résultats pouvant être généralisés nous avions besoin d'un nombre important d'experts évaluant un nombre important de logiciels. En fait, la plupart des experts interrogés ont soulevé le problème de la lourdeur de l'évaluation et considérait implicitement qu'il n'était pas possible d'évaluer plus de 30 logiciels. Pour résoudre ce problème quantitatif, nous avons bâti une méthodologie alternative. Nous avons modélisé le jugement moyen d'expert et nous l'avons utilisé sur un nombre plus important de logiciels. Notre méthodologie a consisté en quatre phases : (étape 1) la définition qualitative de cinq types d'innovation logiciel basé sur un nombre important d'études de cas (environ 300), des entretiens avec des experts et une revue de littérature pertinente ; (étape 2) une évaluation de 25 logiciels libres utilisant ce que nous nommons la *« web-based Delphi »* par 125 experts ; (étape 3) une modélisation conceptuelle de types d'innovation logiciel basé sur le jugement moyen d'experts ; (étape 4) un élargissement de la procédure d'évaluation à 152 logiciels libres. Cette méthodologie peut être résumée par le schéma déjà présenté dans le chapitre méthodologie [Partie I]. Nous vous proposons de détailler chaque étape en mettant en évidence les difficultés et apports de chaque étape.

3. Les apports de l'approche *« web-based Delphi »* : la mobilisation d'experts dans un contexte distribué.

3.1. Etape 1 : la définition qualitative de cinq types d'innovation.

Nous avons théoriquement défini cinq niveaux d'innovation en nous basant sur trois éléments : une étude qualitative exploratoire portant sur près de 300 logiciels, un ensemble d'échanges approfondis avec des experts de l'industrie du logiciel libre et une revue de la littérature pertinente.

Notre étude qualitative exploratoire a porté sur près de 300 logiciels. Pour chaque logiciel libre, nous avons rassemblé des informations à propos : de ses principales fonctionnalités, de ses atouts comparés aux logiciels fermés existants et de son origine. Ces données furent obtenues grâce à l'échange de plusieurs centaines de courriels avec les fondateurs ou les

responsables actuels (au moment des interviews) du développement et d'informations disponibles sur le site officiel de chaque logiciels en triangulant à chaque fois l'information avec d'autres sources (Wikipedia, FreshMeat, forums etc.)

Des interviews approfondies avec des professionnels de l'open source furent menées. Trois experts ont joué un rôle particulièrement fondamental dans la définition des types d'innovation dans le domaine du logiciel libre.

(1) Le premier expert est un chercheur industriel d'Alcatel-Lucent qui *« travaille en R&D dans le monde IP (internet) »* et *« dépose des brevets régulièrement »*. Il est doté d'une expérience de plus de 10 ans en développement et est auteur de plusieurs logiciels libres. Selon lui, l'innovation dans le domaine du logiciel *« est quelque chose : une approche, un procédé qui est nouveau, c'est à dire sans antécédents ([c'est] parfois très subjectif c'est vrai il faut en discuter jusque [devant les] tribunaux), et qui améliore le coût de production, l'efficacité, l'ergonomie, la mise à jour, la fiabilité... il y a même de l'innovation qui ne sert à rien (fonctionnellement) sauf à mieux vendre... »* Cet expert a eu un rôle fondamental dans la définition des types d'innovation et tout au long de notre recherche.

(2) Le second expert est Open Source Architect au sein de Thales D3S (cf. monographie de la [Partie III]). Il est doté d'une expérience de plus de 10 ans en développement. Il occupe des positions de responsabilité dans différents projets européens dédiés au logiciel libre. Il a une longue expérience dans le développement de logiciels libres notamment pour un distributeur de Linux. Nous avons partagé différentes discussions informelles lors de notre participation à un colloque dédié aux technologies open source. Par exemple, cet expert a souligné que *« c'est difficile de se prononcer utilement sur des soft qu'on ne connait pas »* : c'est pour cette raison que nous avons choisi ensuite des logiciels connus afin que ces derniers puissent être évalués.

(3) Le troisième expert est développeur pour une PME éditrice de logiciels libres. Il est également membre du conseil d'administration d'une association de promotion et de défense des logiciels libres. Il a près de dix ans d'expérience en développement de logiciels. Cet expert nous a fait partager son expertise dans le domaine du logiciel libre. Cet experts a eu diverses contributions importantes notamment pour le format des questionnaires envoyés : nous avions commencé par proposer le questionnaire en *« .doc »* mais ce format de fichier (Microsoft) fut reçu comme un manque d'ouverture par certains voire même une provocation. Il a apporté aussi d'intéressantes remarques concernant la sélection des logiciels à évaluer. Sa contribution la plus importante est sans doute la mise en évidence que la hiérarchisation

artificielle des types d'innovations pouvait orienter de manière très importante les résultats des études menées.

Enfin une revue de la littérature ciblée, nous a permis de définir théoriquement cinq types d'innovations. Les cinq types sont les suivants : (1) *l'alternative libre* ou *free alternative*; (2) *l'émulateur* ou *emulator*; (3) *le package* ; (4) *la pièce d'adaptation* ou *adapter*; (5) *l'orienté nouvel usage* ou *new usage oriented*.

(1) L'alternative libre est un logiciel libre alternatif à un logiciel fermé existant. Un grand nombre de logiciels libres furent développés comme des alternatives. C'est le cas du projet GNU (Hars et Ou 2002: 26; Osterloh et Rota 2007: 164; Spiller et Wichmann 2002: 12) commencé par Richard Stallman comme une alternative libre à Unix (Stallman 1983). C'est aussi le cas du noyau Linux (Transmeta Corporation 1998).

(2) L'émulateur est une version émulée d'un matériel ou d'un ensemble d'applications logicielles. Un nombre non négligeable de logiciels libres émulent du matériel. Par exemple, PCSX2 émule le fonctionnement de la Playstation 2. Pour les émulateurs de logiciels, von Krogh et Spaeth ont souligné que *« les spécialistes ont montré que les produits open source [...] émulaient des logiciels existants dans leur conception [...] »* (von Krogh et Spaeth 2007: 247).

(3) Le package est un logiciel regroupant plusieurs logiciels libres en un seul bloc. La plupart du temps ces logiciels libres visent à faciliter l'usage, l'installation et/ou la configuration de plusieurs logiciels libres plutôt complexes. Les distributions GNU/Linux, le package AMP (Apache, MySQL, Perl et PHP) en sont de bons exemples (Deek et McHugh 2007: 4).

(4) La pièce d'adaptation est un logiciel résolvant un problème technique n'ayant jamais été résolu précédemment. Un des premiers logiciels libres créé est un driver. Il fut créé par R. Stallman pour une imprimante graphique dans le but de résoudre des problèmes non résolus auparavant par le pilote Xerox (Hicks et Pachamanova 2007: 36; Stallman 2002).

(5) L'orienté nouvel usage est un logiciel introduisant un nouveau concept d'usage. Par exemple, Mute est un logiciel de Peer to Peer (P2P) permettant à ses utilisateurs de conserver leur anonymat. Mute introduisit la notion d'anonymat si l'on le compare à d'autres logiciels de P2P (Wikipedia 2008). Le mécanisme de routage de Mute fut inspiré de la manière dont les fourmis communiquent lorsqu'elles sont en recherche de nourriture. Habituellement lorsqu'un individu partage des fichiers sur des réseaux de P2P traditionnels chaque fichier est associé avec une adresse IP (Internet Protocol) où le fichier est stocké. Il est très simple pour des institutions constamment à la recherche de violation de droits d'auteurs (comme la bien

connue RIAA) d'identifier les responsables de ces infractions. Mute introduit une nouvelle fonction consistant à associer chaque fichier avec une adresse virtuelle (Mute Team 2008).

3.2. Etape 2 : Test des types d'innovations via l'évaluation d'experts de 25 logiciels libres en utilisant la Web-based Delphi.

3.2.1. Données et profils des experts.

Dans le but de tester la pertinence des catégories dérivées de notre étude qualitative, nous avons transformé les types d'innovation en autant d'affirmations testables en utilisant une variante de la technique Delphi (Rowe et Wright 1999). Pour cela, nous avons tout d'abord créé une liste de 152 logiciels libres. Cette liste fut créée à partir des 100 logiciels les plus populaires à la fois des sites internet SourceForge (100) et FreshMeat (100) cela en suivant la remarque de Spaeth, von Krogh, Sturmer et Haefliger qui ont mis en évidence le problème de la représentativité des études reposant uniquement sur des données provenant de SourceForge (Spaeth et al. 2007: 27). Puis, nous avons choisi cinq logiciels préalablement identifiés dans chaque catégorie, soit 25 logiciels au total (voir le tableau ci-dessous).

Tableau 26 : Les 25 logiciels libres sélectionnés pour l'évaluation.

Nom	Définition	Classification arbitraire
7-Zip	Utilitaire de compression de fichiers.	Alternative libre.
ALSA Driver	Pilote son pour Linux.	Pièce d'adaptation.
BitTorrent	Protocole de partage de fichiers en P2P.	Orienté nouvel usage.
DOSBox	Emulateur Dos.	Emulateur.
FileZilla	Client FTP.	Alternative libre.
Gimp	Programme d'édition d'images.	Alternative libre.
KNOPPIX	Distribution Linux en live CD.	Package.
Linux NTFS	Pilote pour système de fichiers NTFS pour Linux.	Pièce d'adaptation.
MiKTeX	Distribution de TeX.	Package.
MinGW	Adaptation d'outils de développement GNU aux systèmes W32.	Pièce d'adaptation.
Ncurses	Emulation de System V.	Emulateur.
Nmap	Scanner de sécurité.	Orienté nouvel usage.
PDFCreator	Outil de création de fichiers PDF.	Alternative libre.
PeerGuardian	Outil de protection de la vie privée pour le P2P.	Orienté nouvel usage.
PHP	Langage de Script.	Orienté nouvel usage.
Pidgin	Client de messagerie instantané multi-protocole.	Orienté nouvel usage.
PortableApps.com	Package d'applications portables.	Package.
Samba	Système d'interopérabilité entre les systèmes Linux/Unix et Windows.	Pièce d'adaptation.
ScummVM	Emulateur du système Scumm.	Emulateur.
Util-linux	Suite d'utilitaires pour les systèmes Linux.	Package.
VirtualDub	Logiciel de retouche video.	Alternative libre.
VisualBoyAdvance	Emulation de la GameboyAdvance.	Emulateur.
Wine	Implémentation de Windows pour les systèmes Unix/Linux.	Pièce d'adaptation.
XAMPP	Installeur Apache, MySQL, PHP et Perl.	Package.
ZNES	Emulateur de la Super Nintendo.	Emulateur.

Après la création de cette liste, nous avons converti les cinq types d'innovation identifiés en autant d'affirmations testables. En d'autres termes, nous avons créé un questionnaire et nous avons demandé à des *experts* du réseau que nous avons bâti au fur et à mesure de notre recherche pour évaluer ces 25 logiciels. Ces experts furent recrutés lors de différentes interactions avec le terrain : il y a un nombre non négligeable de membres provenant des cas étudiés : OW2 Consortium, OpenOffice.org, Kexi et Freeworks. Beaucoup d'experts ont eux mêmes contactés des personnes de leurs réseaux. Nous avons aussi contacté des associations d'utilisateurs de logiciels libres à travers les quatre continents qui ont véritablement servi de relais en postant un lien vers notre questionnaire mis à disposition en ligne. Au final, cette étude fut menée auprès de spécialistes du logiciel libre provenant de diverses institutions : 70% travaillent dans l'industrie et les services IT (exemple : Thales, Google, Alcatel-Lucent, Motorola etc.) et de centres de recherche (exemple : INRIA, NASA, la Marine Danoise, CNRS, etc.) à travers toute la planète (voir la description des experts ci-dessous).

Tableau 27 : Secteurs d'activités des experts.

Champs	Nombre[1]	Pourcentage
Industrie et services IT	62	59.62%
Universités/Ecoles	22	21.15%
Services (hors IT)	11	10.58%
Centres de recherché	9	8.65%
Total	104	100%

[1] Les données n'étaient pas disponibles pour 21 répondants.

Tableau 28 : Postes occupés par les experts.

Emplois	Nombre[2]	Pourcentage
Développeurs	29	25,22%
Administrateurs systèmes	19	16,52%
Etudiants	18	15,65%
Expert/Architecte/Ingénieur logiciel	18	15,65%
Autres	11	9,57%
Chercheurs (académiques)	8	6,96%
Directeurs Techniques et R&D	7	6,09%
Managers de Projets	3	2,61%
Dirigeants	2	1,74%
Total	115	100,00%

[2] Les données n'étaient pas disponibles pour 10 répondants.

Tableau 29 : Expérience en développement.

Expérience	Nombre	Pourcentage
Plus de 10 ans	41	32,80%
Entre 5 et 10 ans	23	18,40%
Entre 3 et 5 ans	14	11,20%
Entre 1 et 3 ans	13	10,40%
Moins d'un an	5	4,00%
Ne développe pas	29	23,20%
Total	125	100,00%

Tableau 30 : Origine géographique des experts.

Zones géographiques	Nombre[3]	Pourcentage
Europe	74	62,18%
Asie	14	11,76%
Amérique	26	21,85%
Afrique	1	0,84%
Océanie	4	3,36%
Total	119	100,00%

[3] Les données n'étaient pas disponibles pour 6 répondants.

Pour chaque logiciel nous avons demandé aux experts de s'exprimer sur les cinq dimensions déjà mentionnées en utilisant une échelle de Likert à cinq items (*« Je suis tout à*

fait d'accord » ; *« Je suis d'accord »* ; *« Je ne suis pas d'accord »* ; *« Je ne suis pas du tout d'accord »* et enfin *« Je ne sais pas*[236] *»*) :

- Pour les alternatives libres : *Ce logiciel libre est avant tout une alternative libre à un logiciel « propriétaire » (closed source) existant. Exemple : aMSN est un clone libre du client de messagerie MSN Messenger.* / *This open source software is first of all an open source alternative to a main piece of proprietary (closed source) software. Example: aMSN is an open source clone of MSN Messenger the instant messaging client.*
- Pour les émulateurs : *Ce logiciel libre émule le fonctionnement d'un matériel physique ou d'un ensemble d'applications logicielles (ex : système d'exploitation, moteur, etc.) : Exemple : PCSX2 est un logiciel qui émule le fonctionnement de la console Playstation 2.* / *This open source software emulates hardware or a group of applications (e.g.: operating system, engine, etc.). Example: PCSX2 emulates the functioning of Playstation 2 (a games console).*
- Pour les packages : *Ce logiciel libre est principalement un ensemble de logiciels libres. Exemple : EasyPHP réunit Apache, MySQL et PHP en un seul logiciel. / This open source software is mainly a package of open source software. Example: EasyPHP assembles Apache, MySQL and PHP in one package.*
- Pour les pièces d'adaptation : *Ce logiciel libre a résolu un problème technique qui jusqu'ici n'avait pas été résolu. Exemple : Mono est le premier logiciel qui a permis d'exécuter des applications .Net sur un système Linux. / This open source software has solved a technical problem which had not previously been solved. Example: Mono is the first software to make it possible to run .Net applications (Microsoft) on a Linux system.*
- Pour les orientés nouvel usage : *Ce logiciel libre a créé un nouvel usage. Exemple : Mute est un logiciel de P2P (Peer to Peer) permettant de préserver l'anonymat de ses utilisateurs. Mute a ajouté la notion d'anonymat si on le compare aux logiciels de P2P existants lorsqu'il a été initié. / This open source software has created a new usage. Example: Mute is P2P (Peer to Peer) software which makes it possible for users to remain anonymous. Mute has added the notion of "anonymity" if we compare it to other P2P software.*

[236] Il convient de noter que la possibilité *« je ne sais pas »* est très importante dans une recherche portant sur l'évaluation de logiciels car comme cela fut souligné par un spécialiste de l'industrie, tous les experts ne connaissent pas tous les logiciels à évaluer. Il est donc nécessaire d'introduire la possibilité d'exprimer leur non connaissance afin d'éviter tout jugement non utile car non basé sur la connaissance effective du logiciel.

3.2.2. Agrégation des données et traitement statistique.

Nous avons agrégé les 125 évaluations des experts (Rowe et Wright 1999: 354) pour chaque logiciel : soit 3125 évaluations (125*25). Après cela, nous avons appliqué une analyse en composantes principales et une catégorisation statistique afin de tester la pertinence de nos variables et de comparer notre première catégorisation avec celle des experts.

L'Analyse de Composantes Principales (ACP) pour données ordinales confirme le fait que nos variables conduisent à la création de groupes de logiciels libres très proches de notre catégorisation qualitative. L'ACP propose deux axes : le premier explique 33,8% de la variance (Alpha de Cronbach : 0,895) et le second explique 30,29% de la variance (Alpha de Cronbach : 0,879). Les deux axes réunis expliquent 63,68% de la variance (Alpha de Cronbach : 0,970).

Figure 32 : Analyse en Composantes Principales.

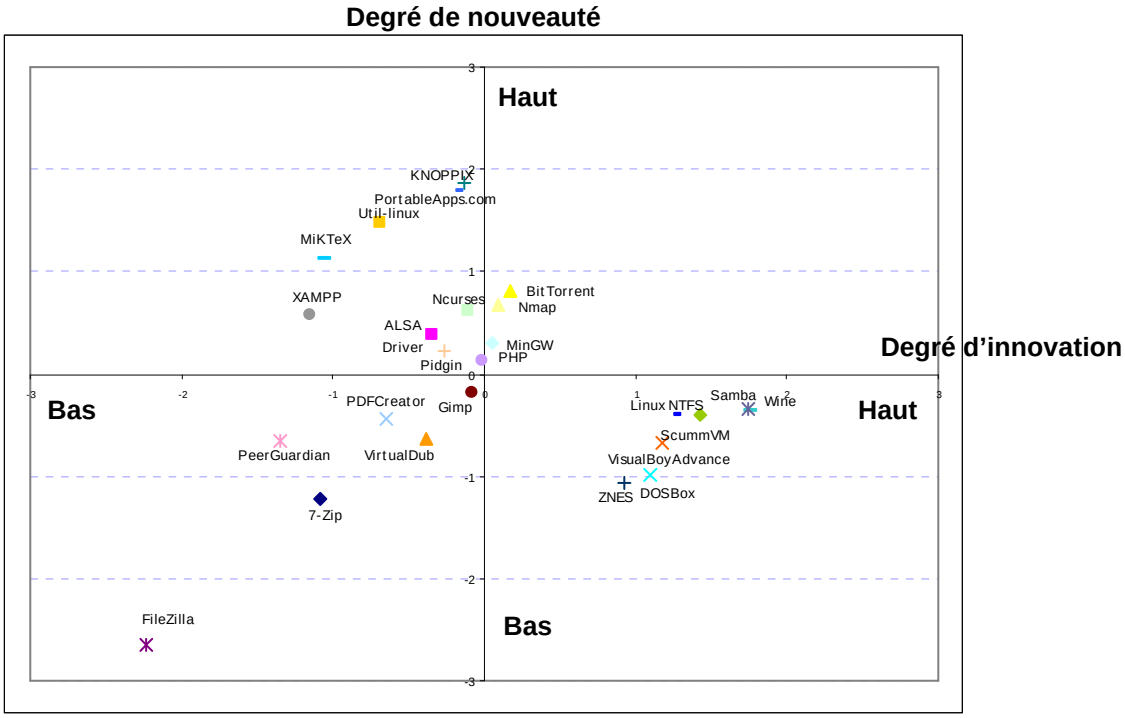


Figure 33 : Analyse en Composantes Principales sur variables.

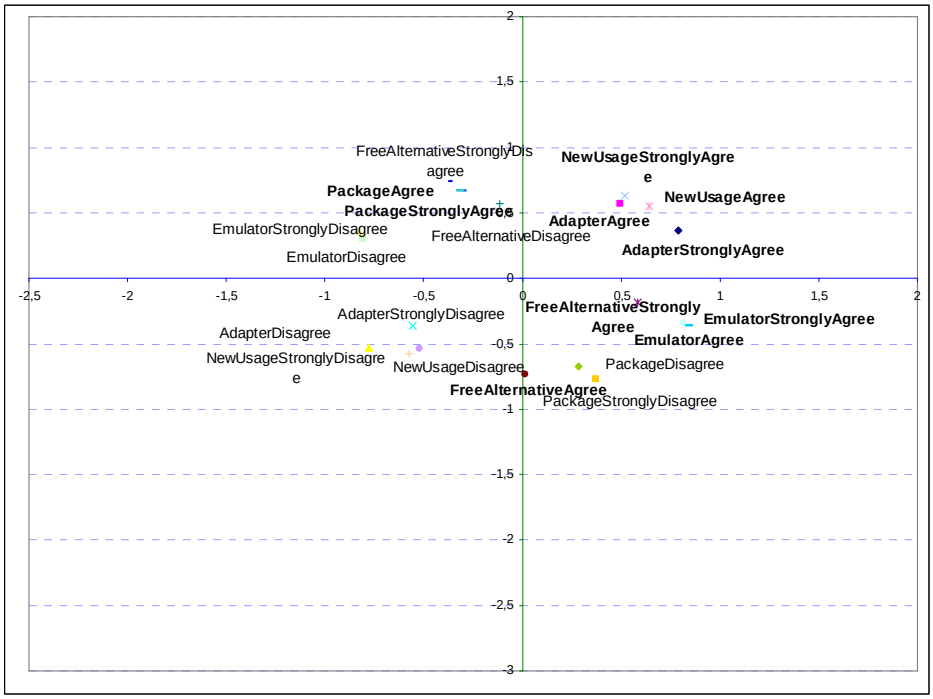


Tableau 31 : Catégorisation qualitative et statistique du premier échantillon.

Logiciels	Type ex-ante	Type ex-post
LinuxNTFS	Adapter.	1 (Adapter).
Samba	Adapter.	1 (Adapter).
ScummVM	Emulator.	1 (Adapter).
Wine	Adapter.	1 (Adapter).
Knoppix	Package.	2 (Package).
MiKTeX	Package.	2 (Package).
MinGW	Adapter.	2 (Package).
PortableApps	Package.	2 (Package).
Utillinux	Package.	2 (Package).
XAMPP	Package.	2 (Package).
ALSADriver	Adapter.	3 (New usage oriented).
BitTorrent	New usage oriented.	3 (New usage oriented).
Ncurses	Emulator.	3 (New usage oriented).
Nmap	New usage oriented.	3 (New usage oriented).
PHP	New usage oriented.	3 (New usage oriented).
DosBox	Emulator.	4 (Emulator).
VisualBoyAdvance	Emulator.	4 (Emulator).
ZNES	Emulator.	4 (Emulator).
7zip	Free Alternative.	5 (Free Alternative).
FileZilla	Free Alternative.	5 (Free Alternative).
Gimp	Free Alternative.	5 (Free Alternative).
PDFCreator	Free Alternative.	5 (Free Alternative).
PeerGuardian	New usage oriented.	5 (Free Alternative).
Pidgin	New usage oriented.	5 (Free Alternative).
VirtualDub	Free alternative.	5 (Free Alternative).

Cluster centers	1	2	3	4	5	Moyenne
AdapterStronglyAgree	0,60	0,32	0,40	0,33	0,19	0,37
AdapterAgree	0,37	0,42	0,41	0,47	0,37	0,40
AdapterDisagree	0,02	0,19	0,17	0,12	0,35	0,17
AdapterStronglyDisagree	0,02	0,07	0,03	0,08	0,10	0,06
Free AlternativeStronglyAgree	0,52	0,20	0,18	0,20	0,38	0,30
Free AlternativeAgree	0,28	0,22	0,20	0,34	0,32	0,27
Free AlternativeDisagree	0,14	0,28	0,37	0,29	0,18	0,25
Free AlternativeStronglyDisagree	0,06	0,30	0,24	0,16	0,12	0,18
EmulatorStronglyAgree	0,40	0,09	0,07	0,55	0,07	0,24
EmulatorAgree	0,31	0,15	0,07	0,32	0,08	0,18
EmulatorDisagree	0,15	0,33	0,36	0,04	0,34	0,24
EmulatorStronglyDisagree	0,15	0,44	0,49	0,09	0,51	0,34
NewUsageStronglyAgree	0,31	0,26	0,30	0,20	0,17	0,25
NewUsageAgree	0,34	0,36	0,40	0,35	0,25	0,34
NewUsageDisagree	0,27	0,27	0,24	0,29	0,41	0,30
NewUsageStronglyDisagree	0,08	0,11	0,05	0,16	0,18	0,12
PackageStronglyAgree	0,10	0,55	0,09	0,08	0,10	0,18
PackageAgree	0,16	0,30	0,15	0,05	0,16	0,16
PackageDisagree	0,38	0,10	0,44	0,41	0,42	0,35
PackageStronglyDisagree	0,36	0,06	0,32	0,45	0,32	0,30

La Catégorisation Statistique (CS) montre deux éléments. Premièrement, la CS calcule des groupes (clusters) proches de notre classification qualitative.

(1) Le groupe 1 (Cluster 1) correspond au *Cluster des pièces d'adaptation (adapters)* car c'est le cluster où la valeur des cluster centers (CC) pour les variables *adapter* et *free alternative* sont les plus hauts (voir tableau ci-dessus). La haute valeur des variables *adapter* est cohérent avec l'hypothèse que les pièces d'adaptation résolvent des problèmes techniques n'ayant pas été précédemment résolus. La haute valeur de la variable *free alternative* est aussi cohérente étant donné que ces logiciels reproduisent un format fermé sous une autre forme.

(2) Le Cluster 2 peut être considéré comme le *Cluster des packages* car c'est le cluster pour lequel la valeur des cluster centers (CC) pour les variables *package* sont les plus hautes. Cela est cohérent avec l'hypothèse que les packages sont principalement un regroupement de quelques logiciels libres.

(3) Le Cluster 3 peut être considéré comme le *Cluster des orientés nouvel usage* car c'est le cluster où la valeur des cluster centers (CC) pour les variables *new usage* sont les plus hautes. Cela est cohérent avec l'hypothèse que les logiciels orientés nouvel usage créent un nouveau concept d'usage.

(4) Le Cluster 4 peut être considéré comme le *Cluster des émulateurs* car c'est le cluster où la valeur des cluster centers (CC) pour la variable *emulation* est la plus haute. Cela est cohérent avec l'hypothèse que les émulateurs émulent du matériel ou un groupe d'applications logicielles.

(5) Le Cluster 5 peut être considéré comme le *Cluster des alternatives libres* car c'est le cluster dans lequel où la valeur des cluster centers (CC) pour les variables *free alternative* sont les plus hautes et où les autres valeurs sont toujours à un niveau inférieur de la moyenne. Cela est cohérent avec l'hypothèse que les alternatives libres sont avant tout des alternatives libres de logiciels fermés dominants.

Deuxièmement, la Catégorisation Statistique (CS) classifie 4/5 (exactement 19 logiciels sur 25) des logiciels initialement classifiés ensemble de la même manière (voir le tableau ci-dessus) Cela confirme bien que notre catégorisation qualitative et nos variables sont pertinentes pour ranger les logiciels par types d'innovation. Toutefois, la CS montre quelques différences entre notre classification qualitative et celle des experts : 6 logiciels sur 25 présentent une différence de classification.

(1) ALSADriver fut considéré (par les experts) comme orienté nouvel usage alors que nous l'avions classé comme pièce d'adaptation. Ce logiciel a certainement été considéré comme orienté nouvel usage car *« Alsa sauve Linux pour le son »* (Expert n°5, INRIA).

(2) MinGW fut classifié comme un package (cela est vrai car MinGW regroupe des outils GNU) alors que nous l'avions rangé dans la catégorie des pièces d'adaptation car il s'agit d'une adaptation des outils GNU pour les systèmes Windows. MinGW signifie : Minimalist GNU pour Windows (MinGW Team 2008).

(3) Ncurses fut considéré comme orienté nouvel usage alors que nous l'avions vu comme un émulateur. En effet, *« la bibliothèque Ncurses (new curses) est une émulation libre des curses de System V »* (Free Software Foundation 2007).

(4) PeerGuardian fut classé comme une alternative libre alors que nous l'avions classifié comme orienté nouvel usage. En revanche, les données montre que ce logiciel n'est globalement pas assez connu par nos experts (66% des experts ne connaissent pas PeerGuardian) ce qui rend la classification très difficile.

(5) Pidgin fut classifié comme alternative libre alors que nous l'avions rangé parmi les orientés nouvel usage. En effet, selon un de nos experts *« Pidgin [est] innovant dans le sens que c'est un tout protocole en un seul logiciel »* (Expert n°2). Pidgin est un client de messagerie instantané multi-protocole. Il fut probablement vu comme une alternative libre aux clients existants (Live Messenger, Yahoo Messenger, AIM, etc.).

(6) ScummVM fut classé comme émulateur alors que nous l'avions vu comme une pièce d'adaptation. Mais il est vrai que *« ScummVM […] est un émulateur du moteur Lucas »* (Expert n°23, University of Melbourne).

Pour résumer, les différences entre notre classification qualitative et celle calculée à partir des évaluations des experts provient d'une raison principale : il existe des logiciels libres entre plusieurs catégories. Néanmoins notre classification qualitative semble être une bonne approximation du jugement moyen des experts.

3.3. Etape 3 : La modélisation de l'innovation logicielle basée sur la moyenne des évaluations des experts.

Grâce aux analyses statistiques réalisées à partir des évaluations d'experts, nous avons dérivé une modélisation des types d'innovation de l'industrie du logiciel. Chaque type peut être associé avec des propriétés dominantes issues des caractéristiques de chaque cluster.

Tableau 32 : Modélisation des types d'innovations.

Types d'innovations.	Propriétés.
C_1 = Alternative libre ↔ $C_1 = P_1$ C_2 = Emulateur ↔ $C_2 = P_2 + P_{4(b)} + P_{5(b)}$ C_3 = Package ↔ $C_3 = P_3 + P_{4(b)} + P_5$ C_4 = Pièce d'adaptation ↔ $C_4 = P_1 + P_2 + P_{4(a)} + P_5$ C_5 = Orienté nouvel usage ↔ $C_5 = P_4 + P_5$	$P_1 = P_{1(a)} + P_{1(b)}$ $P_{1(a)}$ = est fortement une alternative à un logiciel fermé existant. $P_{1(b)}$ = est une alternative à un logiciel fermé existant. $P_2 = P_{2(a)} + P_{2(b)}$ $P_{2(a)}$ = émule fortement un matériel ou logiciel. $P_{2(b)}$ = émule un matériel ou logiciel. $P_3 = P_{3(a)} + P_{3(b)}$ $P_{3(a)}$ = rassemble fortement un ensemble de logiciels libres. $P_{3(b)}$ = rassemble un ensemble de logiciels libres. $P_4 = P_{4(a)} + P_{4(b)}$ $P_{4(a)}$ = résout fortement un problème technique. $P_{4(b)}$ = résout un problème technique. $P_5 = P_{5(a)} + P_{5(b)}$ $P_{5(a)}$ = introduit fortement un nouvel usage. $P_{5(b)}$ = introduit un nouvel usage.

Tableau 33 : Synthétisation des types d'innovations.

Types d'innovations	P1(a)	P1(b)	P2(a)	P2(b)	P3(a)	P3(b)	P4(a)	P4(b)	P5(a)	P5(b)
Alternative libre	1	1	0	0	0	0	0	0	0	0
Emulateur	0	0	1	1	0	0	0	1	0	1
Package	0	0	0	0	1	1	0	1	1	1
Pièce d'adaptation	1	1	1	1	0	0	1	0	1	1
Orienté nouvel usage	0	0	0	0	0	0	1	1	1	1

3.4. Etape 4 : une extension de la procédure d'évaluation à 152 logiciels par une auto-évaluation guidée.

Maintenant, nous pouvons procéder à une auto-évaluation guidée pour un échantillon plus grand de logiciels (152 logiciels). Pour chaque logiciel, nous avons recherché des informations à partir de différentes sources (sites officiels, des courriels échangés avec le(s) auteur(s) de ces logiciels, des échanges avec des experts de l'industrie, les entrées Wikipedia sur le logiciel, des articles etc.) afin de trianguler l'information. Puis nous avons répondu aux mêmes questions que les experts. Enfin, nous avons utilisé les propriétés dominantes du logiciel (voir les types d'innovations décrits ci-dessus) pour choisir le type d'innovation auquel chaque logiciel libre appartenait. Ci-dessous, nous donnons deux exemples de classification guidée par notre grille d'analyse sur les types d'innovations.

Tableau 34 : Exemple 1 : FileZilla, une alternative libre.

Questions	Réponses	Propriétés	Justifications
1) FileZilla est-il fortement une alternative à un logiciel fermé existant ?	Oui.	$P_{1(a)} = 1$	*« Il y a avait déjà beaucoup de clients FTP et nous ne pensions pas être en mesure de vendre une seule copie si nous faisions de FileZilla un logiciel commercial, de ce fait l'open source était la meilleure alternative*[237]*. »*
2) FileZilla est-il une alternative à un logiciel fermé existant ?	Oui.	$P_{1(b)} = 1$	
3) FileZilla émule-t-il fortement du matériel ou du logiciel ?	Non.	$P2_{(a)} = 0$	
4) FileZilla émule-t-il du matériel ou du logiciel ?	Non.	$P_{2(b)} = 0$	
5) FileZilla rassemble-t-il fortement plusieurs logiciels libres ?	Non.	$P3_{(a)} = 0$	
6) FileZilla rassemble-t-il plusieurs logiciels libres ?	Non.	$P3_{(b)} = 0$	
7) FileZilla résout-il fortement un problème technique ?	Non.	$P4_{(a)} = 0$	
8) FileZilla résout-il un problème technique ?	Non.	$P4_{(b)} = 0$	
9) FileZilla introduit-il fortement un nouvel usage ?	Non.	$P5_{(a)} = 0$	
10) FileZilla introduit-il un nouvel usage ?	Non.	$P5_{(b)} = 0$	

[237] Source : Tim Kosse (2008), FileZilla project leader.

Tableau 35 : Exemple 2: Libdvdcss, une pièce d'adaptation.

Questions	Réponses	Propriétés	Justifications
1) Libdvdcss est-il fortement une alternative à un logiciel fermé existant ?	Oui.	$P_{1(a)} = 1$	Libdvdcss est une alternative aux systèmes de cryptages fermés intégrés dans les lecteurs de DVD[238].
2) Libdvdcss est-il une alternative à un logiciel fermé existant ?	Oui.	$P_{1(b)} = 1$	
3) Libdvdcss émule-t-il fortement du matériel ou du logiciel ?	Non.	$P2_{(a)} = 0$	
4) Libdvdcss émule-t-il du matériel ou du logiciel ?	Non.	$P_{2(b)} = 0$	
5) Libdvdcss rassemble-t-il fortement plusieurs logiciels libres ?	Non.	$P3_{(a)} = 0$	
6) Libdvdcss rassemble-t-il plusieurs logiciels libres ?	Non.	$P3_{(b)} = 0$	
7) Libdvdcss résout-il fortement un problème technique ?	Oui.	$P4_{(a)} = 1$	*« Libdvdcss est une simple bibliothèque conçue pour accéder aux DVD en bloc sans avoir à se soucier du décryptage[239]. »*
8) Libdvdcss résout-il un problème technique ?	No.	$P4_{(b)} = 0$	
9) Libdvdcss introduit-il fortement un nouvel usage ?	No.	$P5_{(a)} = 0$	
10) Libdvdcss introduit-il un nouvel usage ?	No.	$P5_{(b)} = 0$	

Cette procédure d'évaluation est une combinaison de ce que nous avons appris de l'évaluation de 25 logiciels libres par les experts et d'informations rassemblées en utilisant les mêmes sources d'informations que dans l'étape 1. Même si avoir 125 experts évaluant 152 logiciels aurait été la procédure idéale. Toutefois, elle est irréaliste pour une raison pratique évidente. Quel expert peut-il connaître 152 logiciels pour apporter un jugement utile ? De plus, la plupart des experts interrogés ont soulevé le problème de la lourdeur du travail d'évaluation. Notre méthodologie nous semble être un bon compromis entre les techniques d'évaluation d'expert et l'auto-évaluation.

L'application de nos critères (comme décrit ci-dessus) à une plus grande liste de logiciels (152) montre que :

(1) Les trois quart des logiciels étudiés sont des alternatives libres à des logiciels fermés existants. Il convient de souligner ici plusieurs éléments. Le fait que la grande majorité des logiciels libres créés par des communautés d'utilisateurs-développeurs soient des alternatives libres ne signifie pas que ces communautés ne sont pas innovantes. Une explication tout à fait intéressante fut apportée par les membres du conseil d'administration de l'AFUL sur les premiers résultats de cette recherche. *« Il est normal qu'une majorité des logiciels soient des clones. Tout simplement parce que les entreprises ne veulent pas investir pour développer sur une plate-forme [Gnu/Linux] qui a une trop petite part de marché (sans parler de pressions probables de vous savez qui) et il faut bien créer le logiciel. Faut-il rappeler le cas de*

[238] Source : (Wikipedia 2009c).
[239] Source : (VideoLAN Project 2008).

l'éditeur Corel, qui ayant commencé à développer pour Linux, a été racheté par Microsoft[240] *? »*

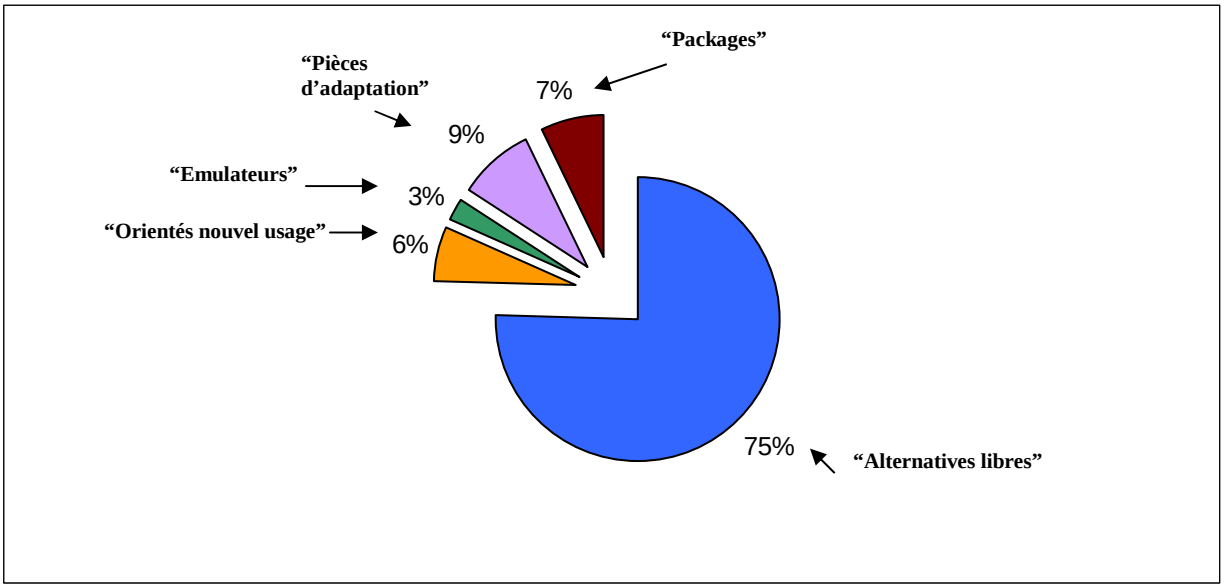
(2) Les utilisateurs-développeurs créent des solutions pour satisfaire leurs besoins n'existant pas sur le marché car cela n'est pas profitable en tant que modèle économique et la plupart du temps résoudre leurs problèmes n'est pas stratégique voire contraire aux intérêts de certaines firmes. Lorsqu'une firme est dominante sur un marché elle n'a aucun intérêt à permettre aux utilisateurs de changer facilement de logiciel. Les utilisateurs-développeurs jouent un rôle particulièrement important dans l'amélioration de l'interopérabilité. Plusieurs exemples furent rencontrés lors de nos investigations : NeoOffice, Samba, Mono, etc.

(3) Les données suggèrent également que certains logiciels sont orientés nouvel usage. Toutefois, ces logiciels semblent concentrés dans trois domaines : la communication (le partage de fichiers, la messagerie, chat, etc.), le développement (langages de script et de programmation) et l'administration de système. D'un côté, ces logiciels semblent être des logiciels pour lesquels les rendements d'adoption (Arthur 1989) sont une partie intégrante de *la valeur du logiciel*, ce qui justifie le fait qu'ils soient distribués librement et gratuitement. Pour certains logiciels le fait de conditionner l'utilisation du logiciel au paiement d'une licence va à l'encontre de la valeur de ce même logiciel. Cela est aussi valable pour les logiciels fermés comme Skype. Plus il y a d'utilisateurs plus le logiciel a un potentiel de valeur. Le cas Firefox est dans une certaine mesure aussi dans ce cas car plus il y a d'utilisateurs plus les possibilités de partenariats sont intéressantes pour les firmes.

D'un autre côté, en nous appuyant sur les analyses de Deek et McHugh, l'open source semble avoir construit lui-même les briques, les outils nécessaires à sa propre expansion (Deek et McHugh 2007: 11). Il n'est donc pas étonnant que la plupart des logiciels innovants du point de vue de l'usage proviennent du domaine de la communication, de la programmation et de l'administration. Selon Deek et McHugh, l'open source est à la fois la cause et le résultat du développement d'internet (Deek et McHugh 2007). Dans le domaine de la communication, Freenet repose également sur le principe du P2P (Peer to Peer) anonyme grâce au recours à la cryptographie tout comme Mute (von Krogh et al. 2003b: 10-11).

[240] Membres du Conseil d'Administration, AFUL, compte-rendu de lecture, 16 juin 2008.

Figure 34 : Extension de l'évaluation à 152 logiciels libres.



4. Discussion et implications : les communautés d'utilisateurs-développeurs ou la compensation des défaillances marchandes.

Le but de cette recherche n'était pas vraiment de savoir si les communautés d'utilisateurs-développeurs étaient une source d'innovation étant donné que la littérature a largement prouvé le caractère innovant des utilisateurs. Le but de cette recherche était plutôt de savoir quels types d'innovations les communautés d'utilisateurs autonomes produisaient. Dans notre étude, nous nous apercevons que les utilisateurs-développeurs créent principalement des logiciels résolvant leurs problèmes. Ces problèmes peuvent être de différentes natures, ce problème peut porter sur : un manque dans un système (alternative libre), un élément technique déjà connu mais non résolu (pièce d'adaptation), une question de configuration ou d'installation (packages) ou encore une création de la version logicielle d'un matériel obsolète abandonnée par son initiateur ou un ensemble d'applications non libres (émulateurs).

Sur la base de ce résultat, nous défendons l'idée que la plupart des innovations utilisateurs dans le domaine de l'open source sont d'une part des améliorations ou des personnalisations de produits (Franke et von Hippel 2003a; Hicks et Pachamanova 2007; Lee et Cole 2003) et d'autre part orientés vers la résolution de problèmes. Cet élément ne semble pas restreint à l'open source puisque la plupart des innovations utilisateurs décrites par von Hippel sont aussi des améliorations de produits ou de la résolution de problèmes (von Hippel 1988), mais aussi celles présentées par Allen dans les hauts-fourneaux (Allen 1983), les machines à vapeur

(Nuvolari 2003), les sports extrêmes (Ulhoi 2004: 1099). Nous pouvons donc raisonnablement conclure que le but des utilisateurs est de satisfaire leurs propres besoins plutôt que d'en définir de nouveaux pour d'autres.

Il convient de souligner que la notion de résolution de problème ne peut être considérée comme un synonyme d'innovation. D'après Nonaka, *« par exemple, l'innovation, qui est une forme clé de la création de connaissances organisationnelles, ne peut être expliquée [de manière] suffisante en termes de traitement de l'information ou de résolution de problèmes. L'innovation peut être mieux comprise comme un processus dans lequel l'organisation créé et définit des problèmes et ensuite développe de nouvelles connaissances pour les résoudre. »* (Nonaka 1994: 14).

Par conséquent, notre recherche ouvre un débat sur l'innovation des logiciels libres conçus par des utilisateurs-développeurs et plus généralement sur tous les produits conçus par des utilisateurs. Les résultats de notre recherche offrent une vision différente de l'open source et des innovations utilisateurs. Elle confirme également l'hypothèse de Christensen dans laquelle il suggérait *« de revisiter les données de von Hippel à la lumière du cadre présenté ici [c'est à dire : le cadre sustaining/disruptive innovation]. Le réseau de valeur pourrait prédire que les innovations à travers lequel les clients dans l'étude de von Hippel conduisent leurs fournisseurs pourraient être des sustaining innovations. Nous pourrions attendre que les disruptive innovations sont parvenues par d'autres sources »* (Christensen 1997: 59).

Il semble que les innovations utilisateurs soient peu adaptées aux contextes fortement incertains comme cela est souligné par une recherche menée chez Securit Saint-Gobain qui avait décidé de se lancer dans un nouvel espace de compétition (le vitrage athermique) alors que la firme était spécialisée dans les formes de vitrages complexes. Les auteurs mettent en évidence que ce cas *« souligne à quel point il ne s'agit ni d'innover avec le client ni d'innover en imitant les concurrents de telles voies sont semées d'embûches, tant le client est versatile, capricieux, incertain… tant il est risqué d'être suiveur si c'est pour être perdant sur les mauvaises pistes et toujours en retard sur les bonnes. »* (Le Masson et al. 2006: 43).

Notre recherche est aussi en accord avec un récent article d'Osterloh et Rota dans lequel ils soulignent que : *« l'innovation utilisateurs est une caractéristique d'une phase d'exploitation d'une technologie, où les innovations sont incrémentales, et non radicales. »* (Osterloh et Rota 2007: 163). En outre, nous ajoutons que MacCormack, Rusnak et Baldwin ont récemment prouvé que les logiciels libres développés de manière distribuée ont une structure plus modulaire que les logiciels fermés (MacCormack et al. 2007: 15) et comme cela

est suggéré par Osterloh et Rota: *« la modularité ne peut être atteinte uniquement si une technologie est bien comprise »* (Osterloh et Rota 2007: 163).

Il est évident que notre travail a des limitations méthodologiques nécessitant d'être améliorées dans des recherches futures. Premièrement, dans la procédure de généralisation il y a un biais de popularité (nous avons analysé des logiciels libres populaires). Deuxièmement, l'évaluation manque de base de comparaison avec des logiciels closed source.

Chapitre 3. LES MODELES D'INNOVATION LIBRES ET FERMES : DE L'OPPOSITION A LA CONTINGENCE.

Dans le chapitre précédent nous avons traité la question de l'innovation sous l'angle du concept général du logiciel. Tuomi disait qu'une *« étude détaillée de l'évolution des projets open source montre qu'ils structurent le processus d'innovation d'une manière très spécifique. »* (Ilkka Tuomi 2005: 436). En utilisant d'autres mots, Tuomi défend l'idée que les *« projets open source »* ont un processus d'innovation particulier, c'est pourquoi nous jugeons nécessaire d'étudier l'innovation non pas du point de vue du concept général du logiciel mais du point de vue applicatif qui semble être le domaine où la spécificité de ce mode de développement apparaît. Nous caractériserons en premier lieu ce que nous nommons *« l'innovation par les extrémités »* [1.]. Ensuite, nous procéderons successivement à une étude des enjeux de la conception de logiciels fermés [2.] et libres [3.].

1. La caractérisation du modèle d'innovation de l'open source : un modèle d'innovation par les extrémités.

1.1. Bases théoriques et empiriques de l'innovation par les extrémités.

Les études de Tuomi ont mis en évidence qu'il y avait une structuration particulière du code source des logiciels libres. Tuomi a montré qu'en fait Linux était structuré en deux : (1) une première partie de code stable changeant très peu (2) une partie de code changeant beaucoup représentant la partie adaptable (Ilkka Tuomi 2005: 437). Tuomi ajoute que *« ce modèle de développement centré sur la communauté, par conséquent, peut être attendu particulièrement compatible avec les innovations et extensions incrémentales qui renforce les valeurs de base de la communauté. Dans ce modèle, il semble bien difficile d'introduire des innovations radicales qui changent la structure courante des pouvoirs ou qui contredit les valeurs centrales de la communauté. »* (Ilkka Tuomi 2005: 439). Nous voyons bien qu'ici Tuomi souligne le fait que la structure sociale des communautés open source et la structure technologique du logiciel conditionnent le type d'innovations générées.

De manière plus simple, les organisations de l'open source ont du mal à gérer les modifications provoquant un trop fort changement dans la structure du logiciel puisqu'à partir du moment où l'on redéfinit l'objet à produire l'innovation par les extensions devient plus

complexe à gérer. Dans ce sens, MacCormack, Rusnak et Baldwin ont récemment prouvé que les logiciels libres développés de manière distribuée ont une structure plus modulaire que les logiciels fermés (MacCormack et al. 2007: 15).

Il semble que pour un même objet de production il est bien difficile pour les organisations de l'open source de gérer à la fois des modifications redéfinissant l'identité[241] du logiciel et les modifications uniquement opérées au niveau des modules. Cela signifie que les communautés d'utilisateurs-développeurs ont bien du mal à être *ambidextres* pour un même produit. Le phénomène du forking est un élément prouvant cette difficulté. Le forking est en effet une scission se produisant lorsqu'il y a des divergences technologiques et/ou idéologiques sur l'évolution d'un logiciel libre. Plus généralement, les logiciels libres semblent structurer l'innovation d'une manière particulière (Ilkka Tuomi 2005: 436).

Selon le Responsable du Centre de Compétences Open Source de Thales, *«* [dans l'open source,] *on a de l'innovation mais dans la chaîne de valeur c'est au niveau applicatif*[242]**.** *»*.

Un autre expert de l'open source a souligné à juste titre : *« Je pense que beaucoup de logiciels ne sont pas innovants lorsqu'ils sont créés, ils le deviennent en ajoutant des fonctions innovantes*[243]*. »*.

Selon le fondateur de Mozilla Europe, *« Bien sûr, il y a de l'innovation aussi dans le monde du logiciel propriétaire, avec des entreprises qui investissent dans le développement de logiciels, mais il y a dans le logiciel libre une autre sorte d'innovation, distribuée et participative. »* (Nitot 2009).

De même, d'intéressants commentaires nous ont été adressés par les membres du conseil d'administration de l'AFUL concernant les premiers résultats de notre étude sur l'innovation (Benkeltoum et Hatchuel 2008) qu'il convient de rendre compte ici. *« L'innovation, ce n'est pas seulement faire quelque chose que personne ne faisait avant. C'est même le cas le plus rare. Souvent l'innovation consiste à faire mieux (techniquement, ergonomiquement, ...) ce que l'on faisait déjà avant. Typiquement Firefox est plein d'innovation, même si l'essentiel de ses fonctionnalités n'est pas nouveau. Et c'est pour cela qu'il peut prendre des parts de marché à Microsoft Internet Explorer*[244]*. »*

Ce qui est mis en évidence dans ces différents propos c'est le caractère massif des innovations incorporées dans certains logiciels libres mais cela se situe au niveau applicatif et

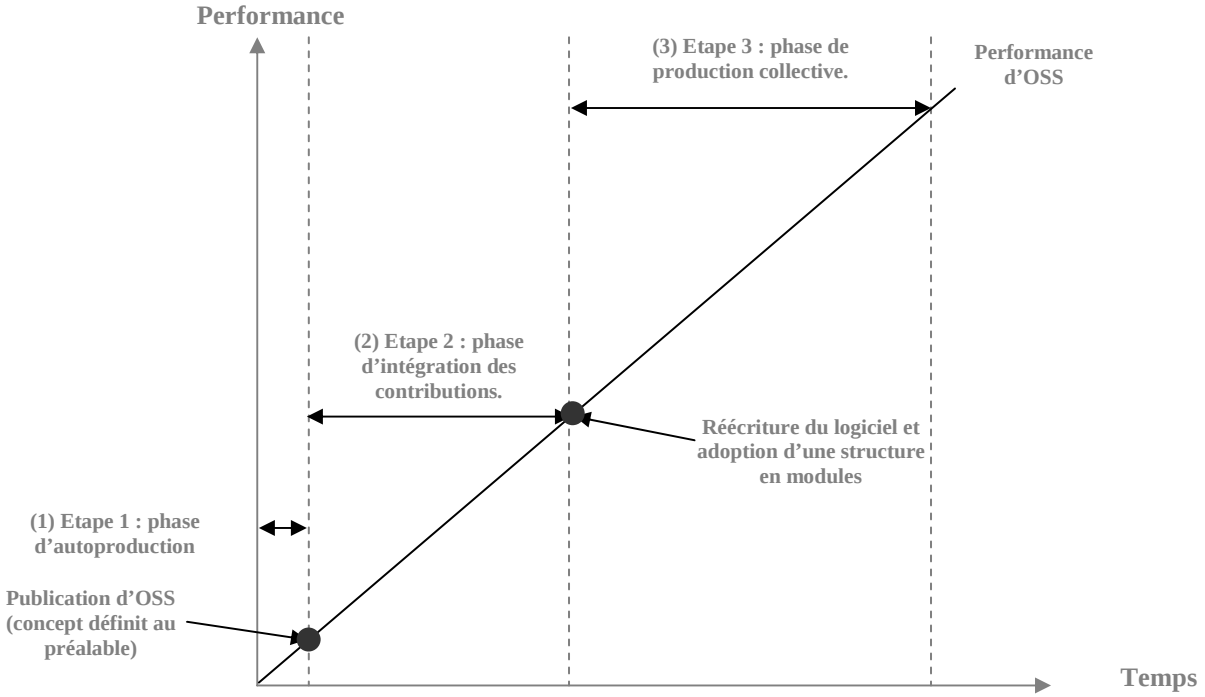
[241] La notion d'identité de produit est empruntée à Le Masson, Weil et Hatchuel (2006).
[242] Responsable du Centre de Compétences Open Source, Thales D3S, entretien en face à face avec l'auteur, 23 janvier 2007.
[243] Expert 2, Responsable de la Sécurité des Systèmes d'Informations (Hébergeur) et Enseignant en Sécurité des Systèmes d'Informations (Ecole ingénieur), Entretien en ligne avec l'auteur, 28 aout 2007.
[244] Membres du Conseil d'Administration, AFUL, compte-rendu de lecture, 16 juin 2008.

non pas au niveau du concept de base du logiciel qui remettrait en question la structure de l'organisation distribuée. Nous pensons qu'il y a un fort couplage entre la structuration du code source, de la communauté et des innovations produites. Nous vous proposons de revenir sur un certains nombre de cas déjà étudiés mais cette fois sous l'angle de la conception afin de vérifier l'hypothèse de l'innovation par les extrémités. Nous défendons l'idée que les logiciels libres développés dans le cadre d'une véritable communauté ont une structure technologique étant à la fois le résultat et la raison de la structure des communautés de l'open source.

1.2. Théorisation de l'histoire du développement des logiciels libres : une structuration technologique et sociale.

Dans le graphique ci-dessous, nous retraçons l'histoire générale d'un logiciel libre réussi. Nous détaillons les différentes phases de développement en mettant en ordonné la performance et en abscisse le temps. Nous étudierons en détail chacune de ces phases en mettant en lumière les éléments théoriques et empiriques soutenant cette modélisation.

Figure 35 : Schématisation de la croissance d'un logiciel libre (OSS).



1.2.1. Etape 1 : phase d'autoproduction ou la prédéfinition du produit à concevoir.

La première phase de développement du logiciel libre (OSS[245] dans le graphique) est une phase d'autoproduction où l'auteur est souvent l'unique utilisateur du logiciel libre. Le logiciel est publié avec un niveau de performance très faible. Il s'agit souvent d'un logiciel qu'il n'est pas possible de mettre en production du fait de son instabilité.

Selon Raymond, il est extrêmement difficile de commencer à partir de rien le développement distribué : il est nécessaire que les développeurs dispersés aient quelque chose pour travailler (Johnson 2002: 660; Raymond 2000: 18). Une étude approfondie de l'origine des logiciels libres aussi bien créés dans un cadre non lucratif de communautés d'utilisateurs-développeurs (Kexi, Freeworks) que dans un but entrepreneurial (Mandriva) signale que le concept général du logiciel est défini à l'avance et qu'il n'est pas discuté ou très peu discuté. Comme le soulignait Armand Hatchuel en réaction à la notion de *« distributed knowledge »* développée par Bruce Kogut et Anca Metiu lors d'un séminaire (Kogut et Metiu 2001b) : *« Bruce parle de « distributed knowledge ». En réalité, dans le projet Linux, il y a un cœur de connaissance, un cœur conceptuel, rédigé de manière solitaire et autoritaire. Ce cœur n'est pas révisable. On a donc une claire opposition, dans ce modèle, entre conception et exécution. »* (Hatchuel 2001: 82).

Les premières versions de ces logiciels étaient très souvent rudimentaires et offraient peu de fonctionnalités (Edwards 2005: 125). Ce qu'apportent les autres contributeurs ce sont des améliorations et des éléments en plus de ce concept. Ce phénomène est aussi valable pour les communautés constituées exclusivement d'entreprises puisque les travaux d'Henkel, à travers la notion de *« jukebox model »*, montre qu'il y a en réalité peu d'interactions dans la conception et que le tout résulte de choix individuels indépendants (Henkel 2004: 18).

Dans la grande majorité des cas, le développement d'un logiciel libre commence par une personne définissant le concept général du logiciel et réalisant les premières ébauches de celui-ci. Plusieurs cas peuvent être présentés pour appuyer cette thèse.

(1) Le noyau Linux a été commencé par Linus Torvalds, il avait déjà défini ce qu'il souhaitait réaliser et cela est clairement spécifié dans son premier message sur Usenet. *«Je suis en train de faire un système d'exploitation (libre) (seulement un hobby, ne sera pas grand et professionnel comme gnu) pour clones AT 386(486). Il était en cours de fabrication depuis avril, et commence à être prêt. »* Linux était au départ quelque chose qu'il n'était pas possible

[245] Open Source Software.

de mettre en production dans une entreprise ou chez un utilisateur final. Pour preuve la première version de Linux était 0.01 cela a une réelle signification dans le domaine du développement informatique. Il désigne une version alpha : c'est-à-dire une version non stable. D'autre part, entre l'annonce du concept (avril 1991) et la première mise à disposition (aout 1991) il y a 5 mois. (2) Fetchmail fut aussi initié par Raymond seul au départ.

Dans les cas étudiés la plupart ont commencé par une seule personne. Selon l'initiateur de (3) FlashMyAdmin (FMA) : *« Concrètement FMA est développé par moi-même au début*[246]*. »*. De même, (4) *« Le projet [Kexi] a été lancé par [Fondateur 1], il y a environ 3 ans*[247]*. »* Enfin l'initiateur de Mandrake (devenu Mandriva) (5) *« J'ai donc travaillé plusieurs mois pour créer Linux-Mandrake, seul et contre tous ;) A la maison. »* (Duval 2005). Nous avons bien vu à travers ces différents cas que les premières versions des logiciels libres étaient très rudimentaires et offraient peu de fonctionnalités.

1.2.2. Etape 2 : phase d'intégration des contributions.

A partir du moment où le logiciel est publié celui-ci entre dans une seconde étape correspondant à une phase d'intégration des contributions centrée sur le concepteur initial du logiciel qui intègrera ou pas les éventuelles modifications apportées (exemple : correction de bugs, ajout de fonctionnalités, etc.). Lors de cette phase d'intégration, le développeur initial est le seul à comprendre l'ensemble du code qu'il a développé et il est le seul en mesure de réaliser l'intégration des contributions externes. Au fur et à mesure, le logiciel améliorera ses performances en étant en quelque sorte tiré par les usages (utilisateurs). Le logiciel libre rattrapera les solutions existantes en termes de fonctionnalités puis il deviendra naturellement compétitif une fois qu'il aura un niveau de performance suffisant face aux logiciels existants (fermés et libres).

Au bout d'une certaine période, variable selon les logiciels libres et l'implication des utilisateurs dans son évolution, le développement du logiciel connaîtra une crise du leadership due à deux phénomènes : d'un côté, les contributeurs auront développé suffisamment de compétences pour proposer de nouvelles fonctionnalités et même devenir responsables d'une ou plusieurs fonction(s) ; d'un autre côté, l'auteur initial souhaitera déléguer (dans le meilleur des cas) une partie du développement aux autres membres de la communauté structurée

[246] Jean-Michel Delettre, Fondateur, Freeworks, Entretien avec l'auteur 3 avril 2006.
[247] Développeur Actif 1, Entretien en ligne avec l'auteur, Kexi, 15 mai 2006.

autour de son logiciel. C'est à partir de ce moment que le logiciel entre dans une nouvelle phase : la production collective.

1.2.3. Etape 3 : phase de mise en production collective.

Le logiciel est réécrit en partie ou en totalité en adoptant une structure en modules désormais indispensable pour la croissance du logiciel car la structure technologique doit être en phase avec la structure sociale de la communauté. Sans cela, il y a un risque de forking : c'est-à-dire la création d'un nouveau logiciel libre sur les bases d'un autre. A la tête de chaque module une *sous-communauté* d'utilisateurs-développeurs se structurera en adoptant le modèle de l'intégration des contributions non pas sur l'ensemble du logiciel comme c'était le cas précédemment mais sur le module dont l'utilisateur-développeur est responsable.

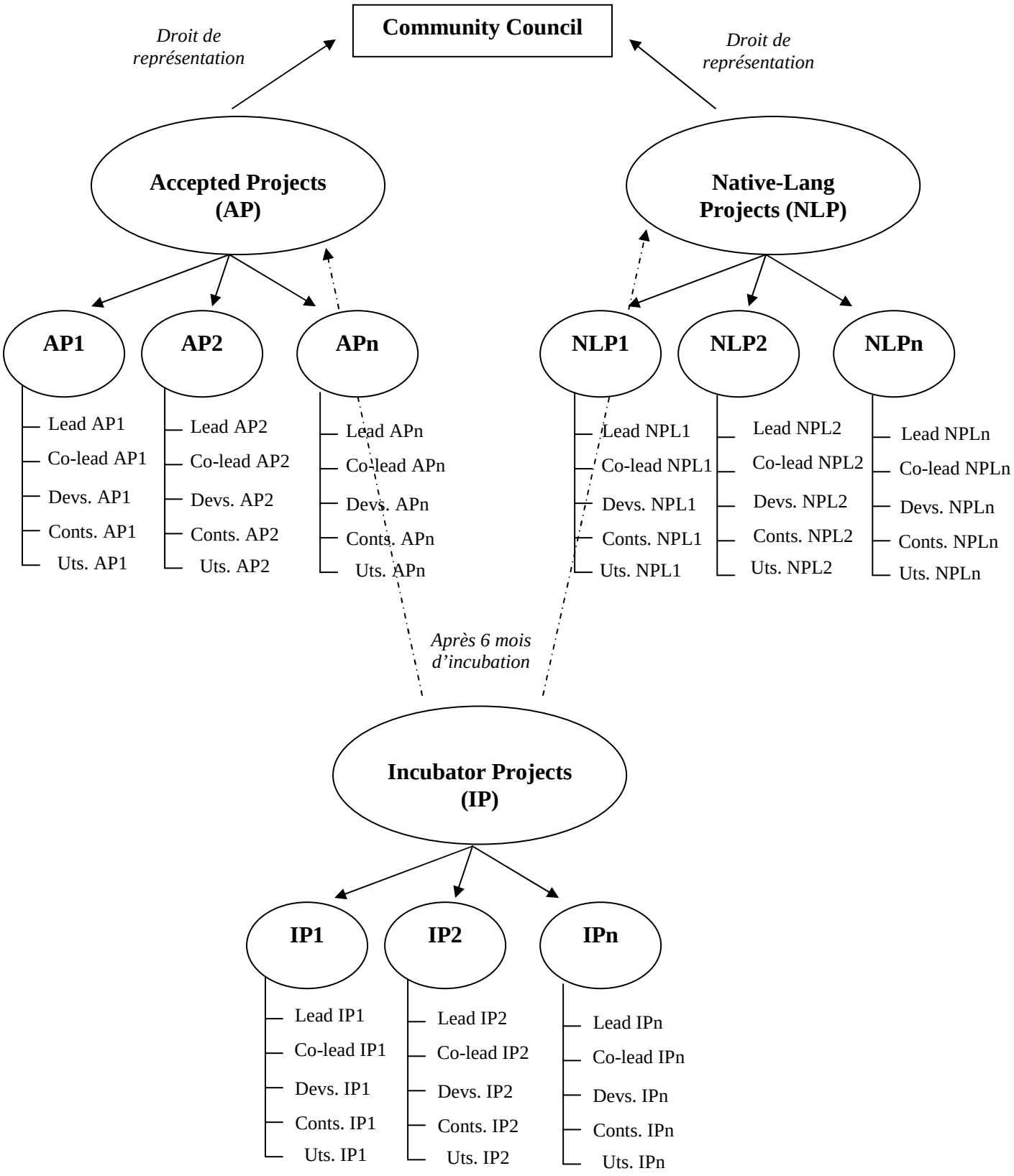
C'est à partir de ce moment que la modularité entrera en jeu. D'après nos analyses, la coordination des acteurs impliqués dans le développement d'un logiciel libre entré dans une véritable production collective est en bonne partie assurée par l'objet de développement : c'est-à-dire le code source constituant un réel instrument de coordination. L'organisation de l'écriture du code source est en lien étroit avec la division du travail dans les organisations de l'industrie du logiciel libre.

Pour ce point, nous nous appuyons sur la littérature existante sur la conception de logiciels. D'après von Krogh et al., la littérature existante sur le développement de logiciels propriétaires suggère que le fait de diviser le code source en modules a trois répercutions bénéfiques sur un projet de logiciel cela : favorise la transparence d'un projet, abaisse les barrières à l'entrée pour contribuer et permet la spécialisation par l'utilisation efficiente des connaissances (von Krogh et al. 2003a: 1218).

Nous avons retrouvé cette volonté générale de simplifier le travail de développement par le biais de l'organisation du code source en modules. Notamment, pour le logiciel FlashMyAdmin de l'association Freeworks pour lequel les développeurs principaux souhaitaient *réécrire* le logiciel en donnant beaucoup d'attention à l'organisation du code source dans le but de favoriser la contribution externe. De même, cette volonté d'organisation du code en classes a également été soulevée par les membres de la communauté Kexi, expliquant que le code source était structuré de manière modulaire afin que chacun puisse s'occuper d'une partie sans se soucier des autres : *« nous travaillons tous sur une partie spécifique de l'application, et jamais sur toute l'application. Excepté peut être [Fondateur 2] »* Développeur Actif 1.

Dans le cas OpenOffice.org (OOo), nous avons clairement une structuration en sous-projets à la fois technologique et humaine. OpenOffice.org est organisé en trois types de *« projets »*. (1) *« Accepted Projects »* : cette catégorie regroupe les objets techniques et ceux concernant le cœur du développement. (2) *« Incubator Projects »* : cette classe rassemble tous les objets devant faire leurs preuves ou des nouvelles idées de développement dont la viabilité n'est pas encore certifiée. Tout nouvel objet de développement passe une période de six mois dans cette catégorie ; une fois que le *« projet »* a montré sa viabilité, il passe soit au statut Accepted Project ou Native-Lang Project. (3) *« Native-Lang Projects »* : le but de ces objets est d'ouvrir un espace de développement et de support dans la langue native des utilisateurs et contributeurs. Cette catégorie est divisée en deux sous parties correspondant au degré d'avancement de l'objet. Le schéma ci-dessous résume ce découpage.

Figure 36 : Les projets d'OpenOffice.org

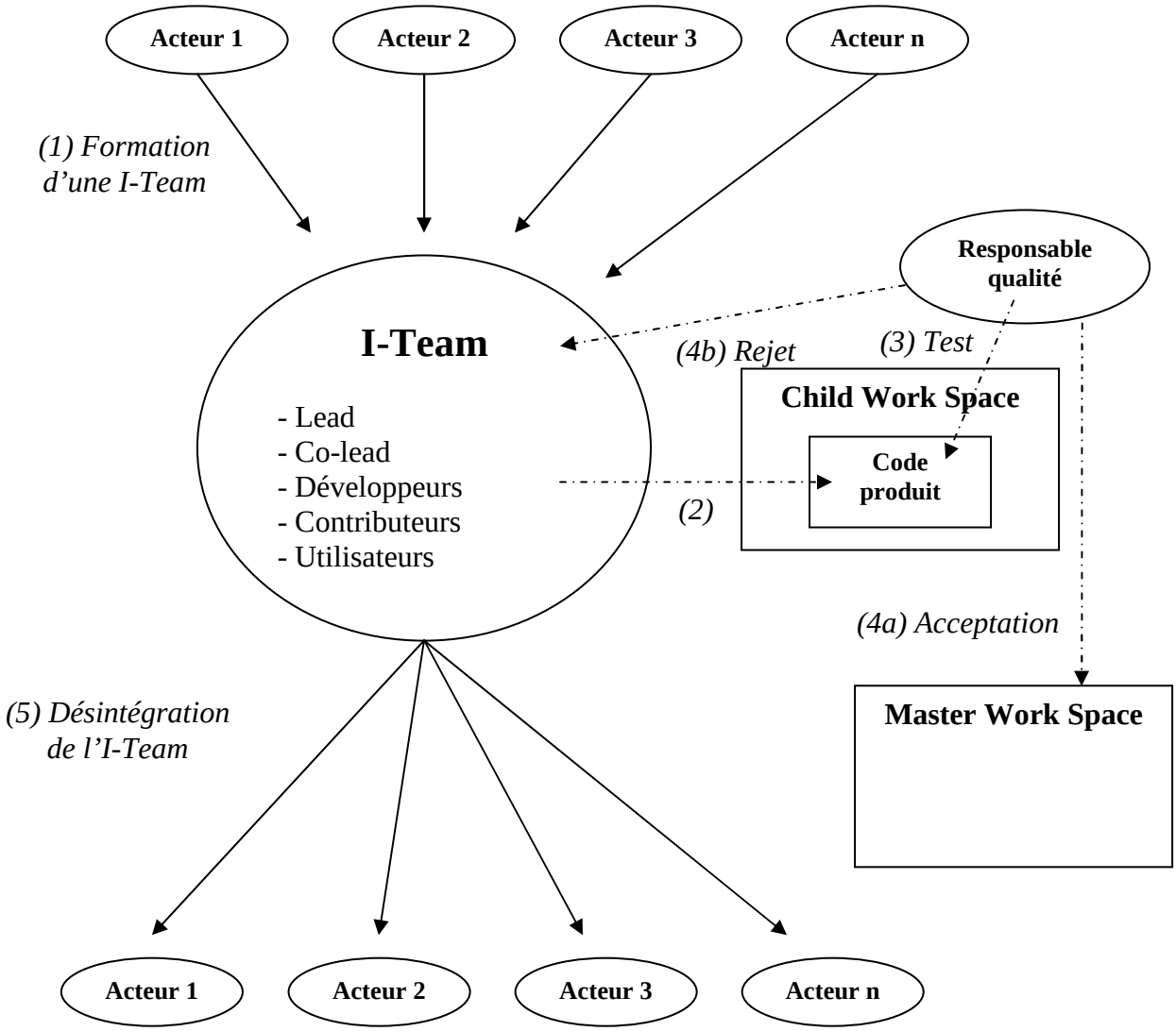


Le développement d'OOo est découpé en sous objets de développement, pour chacun des sous objet, (1) une *« I-Team »* est constituée, dans cette équipe des rôles sont attribués, en plus de ces rôles les acteurs ont des spécialités. Dans chaque équipe, il y a : un lead, un co-lead, des développeurs, des contributeurs et des utilisateurs. La spécialisation des acteurs assure la coopération entre ces derniers. Ainsi, chaque acteur a sa ou ses domaines d'action : l'assurance qualité, le graphisme, le core development, l'usage, etc. Lorsque (2) le code a été produit, celui-ci est (3) testé dans le *« Child Work Space »* par un responsable de la qualité d'OOo. Si (4a) le code est accepté, il sera intégré au code stable dans le *« Master Work Space »*, (5) l'I-Team est alors désintégrée et *« les membres sont assignés à d'autres projets*[248] *»*. Si (4b) le code n'est pas validé, il sera modifié par l'équipe et recommencera le même chemin (2→3) jusqu'à être validé (4a→5).

[248] Membre du Community Council 1, OpenOffice.org.

Pour résumer ce processus, nous vous proposons le schéma ci-dessous.

Figure 37 : Construction et désintégration d'une I-Team.



1.3. Vers une réorganisation du « bazaar »…

Dans la littérature sur l'open source, il est souvent fait référence au mode de développement du *« bazaar »* décrit par Raymond (1999, 2000) comme mode de développement de référence des communautés. Or, nous pensons que les organisations de l'open source sont très structurées contrairement à ce qui est prétendu. Le débat actuel sur la hiérarchie est bien la preuve de ce phénomène.

En matière de génie logiciel, le développement en mode centralisé fonctionne de manière très proche à la manière dont les organisations de l'open source travaillent (The Economist

2006: 65) contrairement à ce que prétendent von Krogh et al. (2003b: 4). Nous citerons quatre citations de spécialistes de l'open source pour appuyer nos propos. La première citation provient du Directeur de la Recherche et de l'Innovation de Thales D3S, *« Les communautés tendent à être très hiérarchisées. Comme par exemple dans Apache, on a trois niveaux : un premier niveau où l'on a le club en gros des fondateurs, les super-committers ; un second niveau où on a les committers : ceux qui ont le droit de déposer du code ; un troisième niveau où on a l'indien de base : le user, ce que je veux dire c'est le développeur de base autorisé à développer du code et à le publier*[249]*. »*

(2) La seconde est issue de l'ancien dirigeant de Mandriva, une entreprise ayant une communauté particulièrement active : *« […] je n'ai pas trouvé d'extraordinaires grandes différences entre le développement en mode libre et le développement en mode centralisé, propriétaire*[250]*… »*

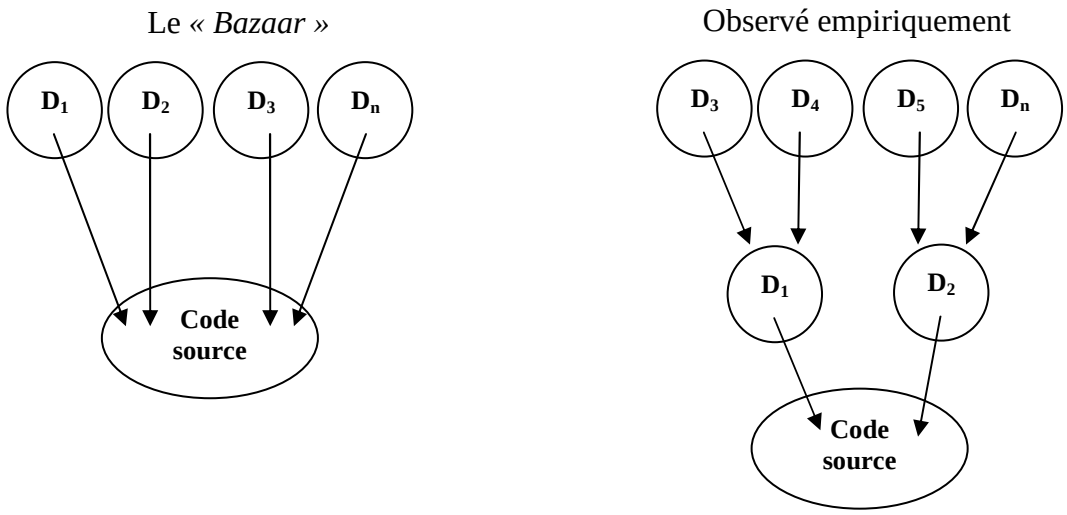
(3) La troisième est issue d'un cadre de Sun Microsystems : « *Il y a une multitude de technologies qui utilisent l'open source pour activer des communautés, avoir des retours, optimiser la notion de contributeurs, mais dans lesquels la conception même du logiciel ne me semble pas totalement éloigné s'il n'avait pas eu de processus open source*[251]*… »*

(4) Et la quatrième du fondateur de Mozilla Europe : *« C'est une communauté [Mozilla] qui est extrêmement structurée. Je […] vais parler de […] la Mozilla Foundation et de Mozilla Europe qui sont […] les projets sur lesquels je travaille quotidiennement depuis quatre ans. […] C'est extrêmement structuré. […] Ah oui, on est tous des anciens de l'industrie informatique*[252]*. […] »*

Pour résumer de manière schématique les deux visions proposées : le modèle du *« bazaar »* et celui qui est effectivement observé sur le terrain, nous vous proposons les représentations graphiques ci-dessous.

[249] Serge Druais, Directeur de la Recherche et de l'Innovation, Thales D3S Shangai, entretien téléphonique avec l'auteur, mercredi 5 aout 2009.
[250] François Bancilhon, ancien Directeur général, Mandriva, entretien en face à face avec l'auteur, mercredi 22 mai 2006.
[251] Technology Advisor, Sun Microsystems, entretien en face à face avec l'auteur, lundi 16 janvier 2007.
[252] Tristan Nitot, Président, Mozilla Europe, Débat sur les logiciels libres et propriétaires, Emission 8-Fi, Direct 8, dimanche 15 mai 2005.

Figure 38 : Le *« bazaar »* réorganisé[253].



Nous nous appuyons également sur les travaux de Loilier et Tellier selon lesquels *« Même si la communauté Linux a été qualifiée de modèle « bazar » (Raymond, 1998), l'analyse de son fonctionnement montre la présence d'unités de pilotage qui ont autorité sur le reste du réseau »* (Loilier et Tellier 2004: 292).

De même, d'autres travaux ont souligné que la répartition du travail entre les acteurs dans les organisations de l'open source est très inégale. Ainsi, seulement quinze personnes contribuent à plus de 88% des lignes ajoutées au code source d'Apache cela signifie qu'une très petite partie du code est réalisée par les autres contributeurs (Mockus, Fielding, et Herbsleb 2000: 5). Ceci fut confirmé dans les travaux portant sur Freenet (von Krogh 2006).

Après avoir mis en avant les spécificités du modèle d'innovation par les extrémités, nous allons nous pencher sur les particularités de la conception de logiciel fermés.

2. Les enjeux de la conception de logiciels fermés.

Nous considérerons deux types de logiciels fermés conçus dans le cadre d'une activité marchande (firme) : (1) les logiciels créant un nouveau marché et (2) les logiciels répondant à un marché existant. Nous verrons à travers deux cas : Photoshop et OfficeOne, que les enjeux de conception du premier et du second type sont très différents.

[253] D désigne Développeur.

2.1. Confrontation de cas empiriques.

2.1.1. Le cas Photoshop.

Par le biais du cas Photoshop, nous prouverons qu'il y a une longue phase réflexion avant la conception d'un logiciel destiné à être commercialisé dans le but de créer un nouveau marché. En effet, entre la première version commercialisée de Photoshop et la première ligne de code développée il y a un décalage de trois années. Durant la première phase, le logiciel avait pour fonction d'afficher des images en noir et blanc, il avait été conçu par Thomas Knoll. John Knoll travaillant à l'époque pour Industrial Light and Magic (ILM), demanda à son frère de travailler avec lui sur un programme lui permettant de retoucher les images (Schewe 2000). Le concept original du logiciel avait bel et bien fait l'objet d'une longue discussion entre les frères Knoll que nous détaillerons dans les prochaines parties.

A. Phase 0 : L'influence de Glenn Knoll[254].

Glenn Knoll, père des frères Knoll, eut un impact important sur les connaissances de Thomas et John Knoll mobilisées plus tard pour créer Photoshop. Glenn Knoll était professeur en sciences radiologiques et en ingénierie nucléaire. G. Knoll était également amateur de photographie et passionné de micro-ordinateur. Si bien que G. Knoll avait sa propre *chambre noire*[255] dans son domicile. C'est par ce biais que John Knoll développa des connaissances à la fois en photographie et en micro-informatique.

John Knoll commença tout d'abord à maîtriser le tirage de clichés en noir et blanc. Puis, il s'intéressa aux techniques de traitement des clichés couleurs nécessitant à l'époque un processus de traitement beaucoup plus complexe qu'aujourd'hui. C'est dans ce contexte que J. Knoll créé ses premières connaissances en matière de manipulation photo couleur. D'un autre côté, J. Knoll était aussi fasciné par l'Apple II Plus récemment acquis par son père. Il s'intéressa à l'informatique à partir de 16 ans. John et Thomas Knoll commencèrent à travailler sur les technologies Apple en 1978 (Apple Computer Inc. 2000).

B. Phase 1 : la naissance de Display.

En 1984, John Knoll réussit à trouver un emploi faisant converger ses deux passions : la photographie et l'informatique. Ainsi, J. Knoll entra au sein d'Industrial Light and Magic (ILM) où il était responsable effets spéciaux et travaillait sur de grosses productions de Lucas

[254] Ce paragraphe est basé sur (Hormby 2007) sauf indication complémentaire.
[255] Une chambre noire est un espace dédié au tirage de photos sur pellicules.

Films (Star Trek, Star Wars, etc.). C'est au sein d'ILM que J. Knoll développa des connaissances pointues en traitement d'image.

Parallèlement à cela, Thomas Knoll débuta une thèse en traitement de l'image. Il travaillait alors sous un Mac Plus ce qui conféra une connaissance pointue des systèmes Mac. A l'époque le Mac Plus n'était pas capable d'afficher les images en niveaux de gris sur un écran, c'est pourquoi T. Knoll créa dans un premier temps un programme permettant d'afficher les dégradés de gris, c'est à ce moment que la première version de *« Display »* a vu le jour. Display n'était qu'un utilitaire en ligne de commande (cela était très courant à ce moment) offrant la possibilité d'afficher des images sous Mac (Hormby 2007).

C. Phase 2 : De l'affichage à l'édition.

En février 86, J. Knoll travaillait chez ILM, en visitant le département chargé du traitement informatisé des images, il assista à une démonstration du Pixar Image Computer (PIC) dont ILM était le premier utilisateur. Le PIC fut mis au point par des employés de Pixar au sein de Lucas Film. Grâce à PIC, il était possible d'extraire une image d'un film à partir d'un scanner, ajouter des effets, la corriger, etc. et la réintégrer dans le film. Aujourd'hui cela peut sembler basique mais à pendant cette période c'était une énorme avancée.

Le travail de thèse de T. Knoll portait sur la manière dont un ordinateur était capable de reconnaitre un objet prédéfini dans une image numérique. T. Knoll avait écrit tout une batterie d'outils permettant le traitement d'images. Lors d'une visite auprès de son frère, J. Knoll fut surpris par la similarité qu'il y a avait entre le travail doctoral de son frère et la machine de Pixar. La différence de prix était telle qu'un PIC coûtait 135 000 dollars alors que Display tournait sur un Mac coûtant 2599 dollars soit 51 fois moins cher.

C'est à ce moment que J. Knoll incita son frère à faire de Display un véritable éditeur d'images permettant aux utilisateurs de manipuler des images numériques comme s'ils étaient dans une chambre noire. John et Thomas commencèrent à travailler ensemble sur le remplaçant de *« Display »* (Hormby 2007).

D. Phase 3 : la coopération entre les frères Knoll ou la création d'ImagePro.

L'interaction entre les frères Knoll provoqua une telle émulation que de nombreux changement furent réalisés sur Display. C'est par ce biais que les fonctions d'éditions furent ajoutées à ce qui n'était qu'un utilitaire d'affichage d'image. John Knoll suggéra d'incorporer

la correction gamma[256] pour que les images affichées à l'écran soient moins sombres. En 1988, Display fut renommé *« ImagePro »*. Ce changement de nom traduit un changement de direction du logiciel.

J. Knoll proposa à son frère de faire d'ImagePro un produit commercial alors que son frère le distribuait gratuitement. Face à la difficulté de doter ImagePro d'un niveau de fonctionnalités suffisant pour en faire un produit commercial, Thomas Knoll décida de suspendre son travail de thèse pendant six mois pour se concentrer sur le développement à plein temps d'ImagePro.

A l'époque, il existait déjà des logiciels permettant l'édition de photo sous Mac et d'autres machines. De ce fait, les frères Knoll eurent des difficultés pour trouver un distributeur pour ImagePro. Diverses firmes furent approchées mais au final, BarneyScan, une firme proposant des scanners a adopté ImagePro qu'elle distribuait avec ses scanners. ImagePro n'était donc pas valorisé en tant que tel mais il accompagnait du matériel comme un utilitaire de ce dernier. Pour des raisons de conflits de propriété de marque ImagePro fut renommé Photoshop au moment où il fut distribué par BarneyScan (Hormby 2007).

E. Phase 4 : La coopération des frères Knoll et Adobe.

Bien que Photoshop fut distribué, les frères Knoll avaient simplement conclu un accord de cession de licence avec BarneyScan donc ils détenaient toujours les droits sur Photoshop. En septembre 1988, les frères Knoll proposèrent Photoshop à Adobe dirigé par Russel Brown qui l'adopta. Au moment où les frères Knoll apportèrent Photoshop, Adobe avait déjà conçu le produit Illustrator (University of Michigan 1998-1999). Photoshop s'associait bien avec ce produit. Adobe mobilisa 15 à 20 développeurs en plus de T. Knoll pour améliorer Photoshop.

En 1990, la première version de Photoshop commerciale fut lancée. Parallèlement à cette mise en commercialisation Photoshop était utilisée par John Knoll au sein d'ILM. Pendant le développement, J. Knoll changea de département à ILM : il passa des effets spéciaux au département de graphisme informatisé. Ce passage eut un impact très important sur l'évolution du programme puisque cela a permis à J. Knoll de confronter le programme aux exigences professionnelles et aussi de connaître quelles fonctions étaient nécessaires pour avoir un produit à un niveau professionnel (Masson 1999).

[256] La correction gamma est une technique permettant d'améliorer la reproduction de l'intensité lumineuse des objets pour les écrans à tubes cathodiques notamment (Potier et Vercken 2000: 1).

2.1.2. Le cas OfficeOne.

Nous présenterons le cas OfficeOne bien qu'Issendis ait utilisé OpenOffice.org pour créer son pack logiciel. Le produit OfficeOne est un pack logiciel conçu pour répondre à une demande existante sur le marché grand public. D'après un des Fondateurs : *« Les deux fondateurs, Didier Urban et moi-même, on a créé la société en 1993. On vient du domaine de la gestion et on a conçu un logiciel dans le domaine de l'archivage. Et c'est à ce moment là archivant des documents : traitement texte, tableur, etc. ; certains de nos clients nous ont dit : « votre produit, il est très bien, il manque plus que de pouvoir produire des documents bureautique avec, et on peut produire et on peut archiver et la boucle est bouclée. »*

A ce moment le marché de la bureautique était largement dominé par les produits Microsoft. Le but d'Issendis *« sur la partie OfficeOne, c'est d'apporter l'alternative à des solutions purement propriétaire Microsoft, qui sont à des prix difficilement accessibles pour tout un chacun. Pour nous c'est pouvoir apporter une solution pour l'utilisateur final. Pour qu'il puisse produire des documents : traitement de texte, tableur, générer du PDF, d'utiliser sa machine dans les meilleures conditions et au meilleur prix. »*

Nous voyons bien que dans le cas OfficeOne il n'y a pas eu de profonde réflexion ni sur le type de marché à cibler ni sur les technologies à mobiliser. L'objectif de la firme était d'offrir un produit compétitif.

2.2. Théorie de la conception du logiciel.

2.2.1. Le logiciel fermé créant un nouveau marché.

Le logiciel fermé créant un nouveau marché fait l'objet d'une longue réflexion dans la phase de définition du produit à concevoir. En se basant sur le cas SynOptics, Chesbrough souligne que le succès de la commercialisation d'une nouvelle technologie dépend de la gestion d'une double incertitude : technique et de marché (Chesbrough 2003: 11). Du point de vue technologique, la firme ne sait pas quelle technologie elle devra mobiliser. Du point de vue marchand, elle ne sait pas quel marché cibler. En somme, elle devra trouver le produit[257] (S_1) tel que S_1 (T ; M). On notera que T est M sont dépendants. Il y a deux cas de figure pour cela : soit T créé un nouvel espace de M ou soit M demande une nouvelle T.

[257] Où S = Logiciel ; T = Technologie ; M = Marché.

La firme devra explorer un ensemble tel que[258] :

$[S_1 (T_1 ; M_1) ; S_2 (T_2 ; M_2) ; S_3 (T_3 ; M_3) \ldots S_n (T_n ; M_n)]$

Où :

$S_1 (T_1 ; M_1) \rightarrow P_1 > 0$?
$S_2 (T_2 ; M_2) \rightarrow P_2 > 0$?
$S_3 (T_3 ; M_3) \rightarrow P_3 > 0$?
...

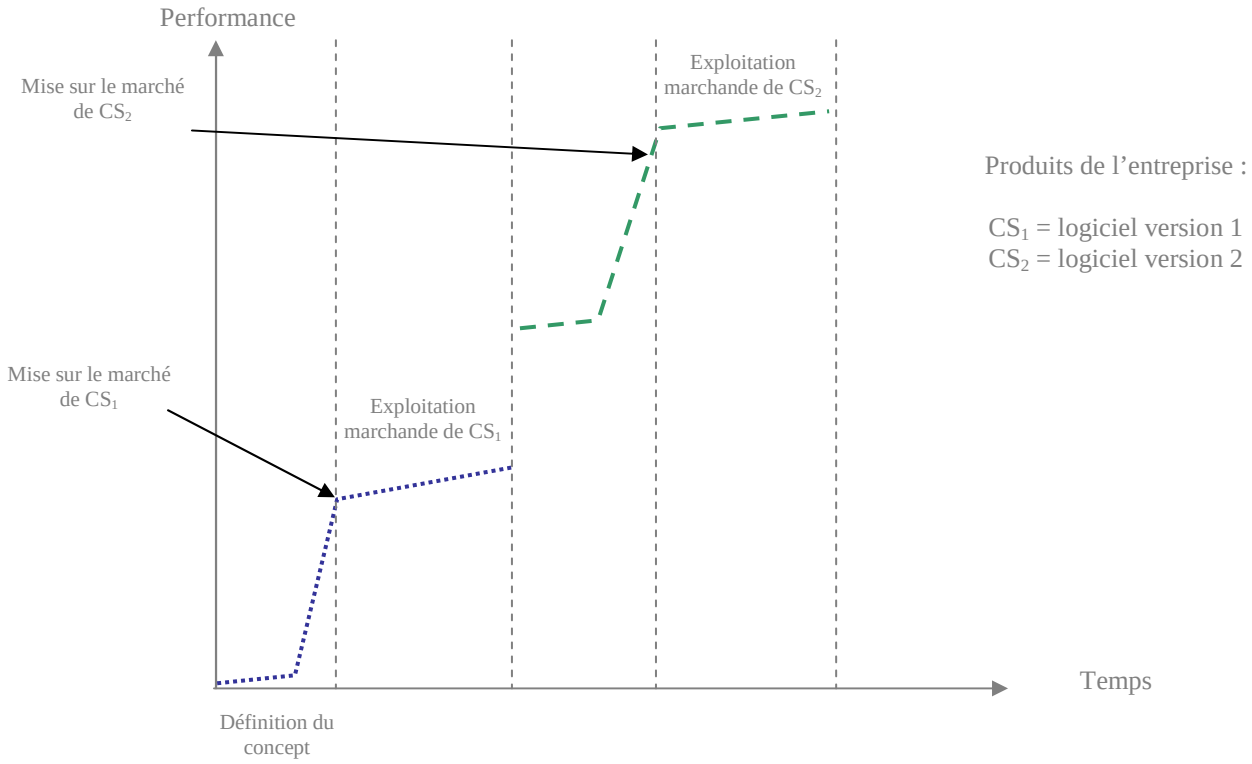
L'inconnue porte à la fois sur T, M et P. La firme ne sait pas quelle technologie elle peut mobiliser, ni quel marché elle devra choisir et encore moins quel profit elle pourra générer. La présente équation nécessite une double exploration : (1) au niveau des technologies et (2) au niveau des marchés.

Le plus souvent un éditeur de logiciel essayera de se positionner sur un créneau inexploité par le marché. Typiquement, il tentera de se créer un monopole ou une niche. Cette étape est particulièrement importante puisque le modèle économique de la firme repose sur cela. L'exemple de Photoshop signale bien qu'il y a une réelle et longue phase réflexion avant la création d'un logiciel destiné à être commercialisé sur un nouveau marché.

Dans le graphique ci-dessous nous modélisons la logique d'exploitation du logiciel fermé.

[258] P = Profit.

Figure 39 : La logique d'exploitation d'un logiciel fermé[259].



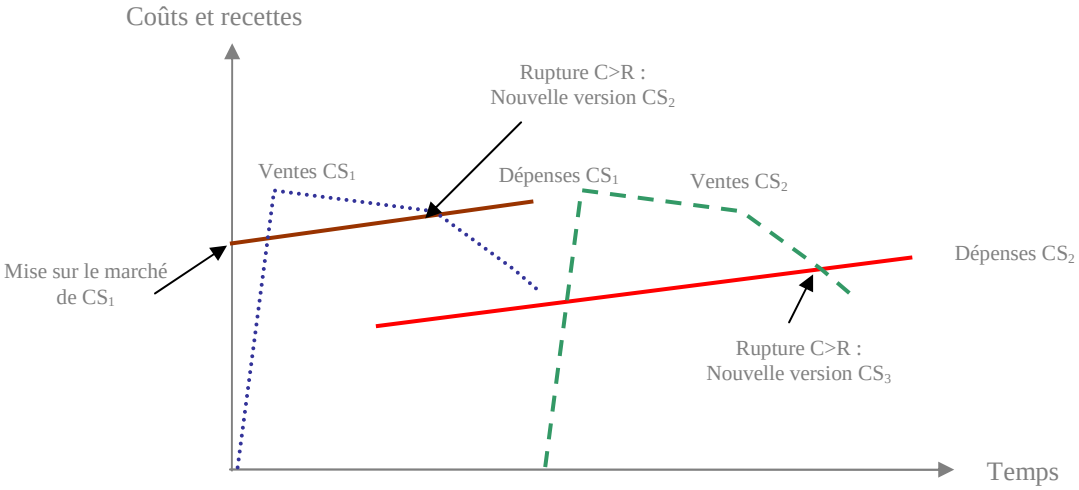
L'innovation incrémentale n'est pas très importante car elle n'est pas source de valeur. L'entreprise préfère incorporer ses innovations dans une nouvelle version comme cela est représenté dans le graphique ci-dessus.

Selon le Directeur de la Recherche et de l'Innovation de Thales D3S, *« Même des acteurs comme IBM à une époque, sur les marchés où ils étaient leader, avaient tendance à sortir au moment voulu les innovations que la firme avait conçu au préalable. Le leader n'a aucune raison d'être innovant. C'est quand le concurrent sort quelque chose d'innovant que le leader fait les deux pas de plus. Ce qui est recherché par le leader c'est la maximisation du profit. On n'est pas dans la recherche de gain de parts de marché donc la principale préoccupation n'est pas dans la recherche de la satisfaction de l'utilisateur[260]. »*

Du point de vue des coûts, la firme attendra soit qu'un concurrent lance un produit soit qu'il y ait une rupture dans les coûts de production comme schématisé dans le graphique ci-dessous.

[259] CS dans le graphique signifie Closed Source
[260] Serge Druais, Directeur de la Recherche et de l'Innovation, Thales D3S, entretien téléphonique avec l'auteur, 5 aout 2009.

Figure 40 : Stratégie de « *versionning* » vue par les coûts[261].



Dans le domaine de l'édition classique, la rente se fait au début de la vie du logiciel. L'éditeur n'a pas intérêt à développer des fonctionnalités nouvelles et à les incorporer automatiquement car ses coûts de production augmenteraient de manière significative sans qu'il y ait une compensation financière du côté du marché. Par conséquent, l'éditeur préfèrera intégrer les nouvelles fonctionnalités dans une nouvelle version du logiciel qu'il lancera de nouveau sur le marché.

2.3. Le logiciel fermé répondant à une demande.

Le logiciel fermé répondant à une demande existante sur un marché fait l'objet d'une réflexion conceptuelle relativement courte en termes de technologie et de marché puisque la demande est connue à priori. Il y a peu d'incertitude technique car les connaissances à mobiliser ont déjà été explorées par les « *firsts movers* » (Christensen 1997: 113) ou premiers arrivants sur le marché. Du point de vue technique, la firme a plutôt à réaliser une fonction de choix sur un intervalle de technologies existantes que d'autres firmes utilisent. Du point de vue du marché, l'étude des solutions concurrentes et des utilisateurs suffira à la firme pour développer un produit alternatif.

La firme devra réaliser un choix entre les technologies utilisées par les produits existants sur un même marché :

[261] C = Coûts ; R = Rentes. Le « *versionning* » est la gestion des versions d'un logiciel.

$[S_1(T_1 ; M_1) ; S_2(T_2 ; M_1) ; S_3(T_3 ; M_1) ; S_4(T_4 ; M_1)]$

Leur nombre est connu et clairement identifié et ils ciblent toujours le même marché. L'incertitude auquel la firme sera confrontée portera essentiellement sur le niveau de rentabilité grâce au nouveau produit.

3. Les enjeux de la conception de logiciels libres.

Dans la section précédente, nous nous sommes intéressés aux enjeux de la conception de logiciel fermés analysés par le biais de la dynamique entre technologie et marché. Dans la présente section, nous nous intéresserons au cas du logiciel libre aussi bien conçu par une communauté d'utilisateurs-développeurs que par une entreprise. Etant donné que nous avons déjà étudié un grand nombre de cas dans le domaine du logiciel libre, nous analyserons directement les enjeux de la conception en faisait références à des cas déjà présentés, au besoin le lecteur intéressé pourra se référer aux monographies correspondantes présentées dans les autres parties de cette thèse.

3.1. Le logiciel libre créé par une communauté d'utilisateurs-développeurs.

Pour le logiciel libre créé par une communauté d'utilisateurs-développeurs, la situation est bien différente puisqu'il ne s'agit pas de trouver un marché pour le produit. Il n'y a donc pas d'incertitude sur le type de marché à cibler puisque par définition ce type de logiciels libres n'est pas conçu pour être vendu mais utilisé. Plusieurs développeurs de logiciels libres ont souligné ce point. Nous citerons quelques extraits d'entretiens.

« Les logiciels opensources répondent toujours à un besoin, au moins celui de son premier utilisateur, son développeur. Un logiciel commercial est destiné à être vendu, pas à servir[262]*. »*

« Je pense que chaque logiciel doit d'une manière ou d'une autre répondre au besoin d'un utilisateur et peut être l'utilisateur seulement lui-même développeur... Il n'y a pas de raison de construire un logiciel qui ne le fait pas à mon humble avis[263]*. »*

[262] Expert 2, Responsable de la Sécurité des Systèmes d'Informations (Hébergeur) et Enseignant en Sécurité des Systèmes d'Informations (Ecole ingénieur), Entretien en ligne avec l'auteur, 27 aout 2007.

« La plupart des logiciels sont créés pour répondre à un besoin existant ! Si ce n'est pas le besoin du créateur, c'est celui d'autres[264]*. »*

Dans le cas du logiciel libre créé par une communauté d'utilisateurs-développeurs, il s'agit de trouver la technologie permettant de répondre à un ou plusieurs besoin(s) ou problème(s) rencontré(s) par un utilisateur. Le développeur devra explorer les technologies possibles pour résoudre le problème rencontré ou répondre au besoin[265].

$[S_1(T_1) ; S_2(T_2) ; S_3(T_3) ; S_n(T_n)] \rightarrow U$

L'inconnue réside uniquement sur la technologie à mobiliser mais le problème (problème à proprement parler ou besoin) est connu donc résoluble ou non. Le développeur explorera la liste des *T* possibles, une fois trouvé il suffira de programmer.

De nombreux logiciels libres peuvent être cités. Le cas NeoOffice illustre bien ce phénomène de résolution de problème. NeoOffice est né de la volonté de certains développeurs d'OpenOffice.org (OOo) de porter la suite bureautique sur le système Mac. Par conséquent le problème était connu.

Soit : U = Créer une version OOo nativement compatible avec Mac OS X.

L'exploration des possibilités par les développeurs d'OOo a montré qu'il existait au moins deux possibilités technologiques.

- T_1 = NeoOffice/J utilisation de Java pour *« modéliser de manière interne ce que OOo était en train de faire à ce que OS X attendait via Java. »*
- T_2 = NeoOffice/C : consistait à *« faire ce que était en train de faire NeoOffice/J, mais en utilisant Cocoa. »* (Drukenbatman 2005).

Soit pour simplifier :

$[S_1(T_1) ; S_2(T_2)] \rightarrow U$

[263] Expert 27, Développeur (Freelance) et Membre actif d'une communauté (gestionnaire de bases données), Entretien en ligne avec l'auteur, 30 aout 2007

[264] Expert 123, Directeur Technique, Site de vente de musique en ligne, Entretien en ligne avec l'auteur, 29 octobre 2007.

[265] Où S = Logiciel ; T = Technologie ; U = Le besoin ou le problème rencontré par l'utilisateur. Dans de nombreux cas le développeur et l'utilisateur peuvent être confondus.

La seconde méthode a été adoptée par les développeurs de NeoOffice (T_2). Il s'agissait ensuite de développer le port alors que la communauté OOo avait initialement choisi (T_1) ce qui a provoqué un fork.

3.2. Le logiciel libre créé par une entreprise.

Dans le cas d'un logiciel libre créé par une entreprise le logiciel peut être dans l'une de ces trois situations : (1) le logiciel vise à remplacer une solution non libre existante, (2) le logiciel est créé pour être vendu (logiciel fermé) et (3) le logiciel est conçu pour être libre.

3.2.1. Le logiciel libre visant à remplacer une solution non libre existante.

Il s'agit d'une situation très proche du logiciel libre créé par une communauté d'utilisateurs-développeurs. Etant donné que le problème est connu il s'agit toujours de résoudre :

$[S_1(T_1) ; S_2(T_2) ; S_3(T_3) ; S_n(T_n)] \rightarrow U$

Par exemple, Thalix a été développé pour remplacer un ensemble de technologies fermées (Stratus) sur lesquelles Thales avait peu de contrôle. C'est aussi ce cas lorsqu'une firme veut créer un composant standard.

3.2.2. Le logiciel libre initialement créé pour être vendu en tant que logiciel fermé.

La firme a échoué à trouver une solution satisfaisante à la double incertitude à laquelle elle était confrontée. Elle disposait donc d'un logiciel aux propriétés suivantes :

$S_1(T_1 ; M_1) \rightarrow P_1 \leq 0$

La firme a échoué à une valorisation directe du logiciel. Par conséquent, elle tentera de libérer le logiciel pour le valoriser indirectement par le biais de produits ou services associés.

Le cas StarOffice était un logiciel sur un marché de la bureautique largement dominé par Microsoft Office. Si StarOffice avait dominé le marché il n'aurait certainement pas été libéré en open source. D'autres cas prouvent le phénomène inverse, pour quelle raison Photoshop,

Norton Antivirus ne sont pas libérés ? Tout simplement car ils sont rentables. Il serait bien étonnant de trouver un logiciel fermé rentable volontairement libéré.

Les libérations de logiciels par des firmes sont de véritables stratégies comme le souligne l'ex-Directeur général de Mandriva : *« Il y a pas mal de gens pour qui le logiciel libre c'est une pure stratégie commerciale, si on regarde des boîtes comme MySQL ou JBoss, l'open source ils s'assoient dessus quand même… Ils s'en foutent complètement, c'est juste une stratégie commerciale. »*

La motivation principale des firmes dans l'open source est économique comme cela fut déjà démontré (Bonaccorsi et Rossi 2003b: 23) contrairement à certains développeurs pouvant être engagés dans le logiciel libre par philosophie.

3.2.3. Le logiciel conçu à la base pour être libre.

Dans ce cas, les firmes peuvent avoir deux types de stratégies. (3a) La firme essayera de valoriser indirectement le produit par le biais de services associés à celui-ci (développements spécifiques, maintenance, installation, etc.). La firme est donc face à une situation incertaine similaire (pour la partie Marché) à celle à laquelle la firme souhaitant lancer un produit fermé sur un nouveau marché. Néanmoins, il convient de souligner ici que la firme ne recherche pas forcément à créer un nouveau marché pour son produit. Par conséquent, le fait d'avoir une nouvelle proposition de valeur n'est pas l'élément central de la firme. La firme incitera à utiliser son produit car cela augmente son potentiel de vente de produits et services associés[266].

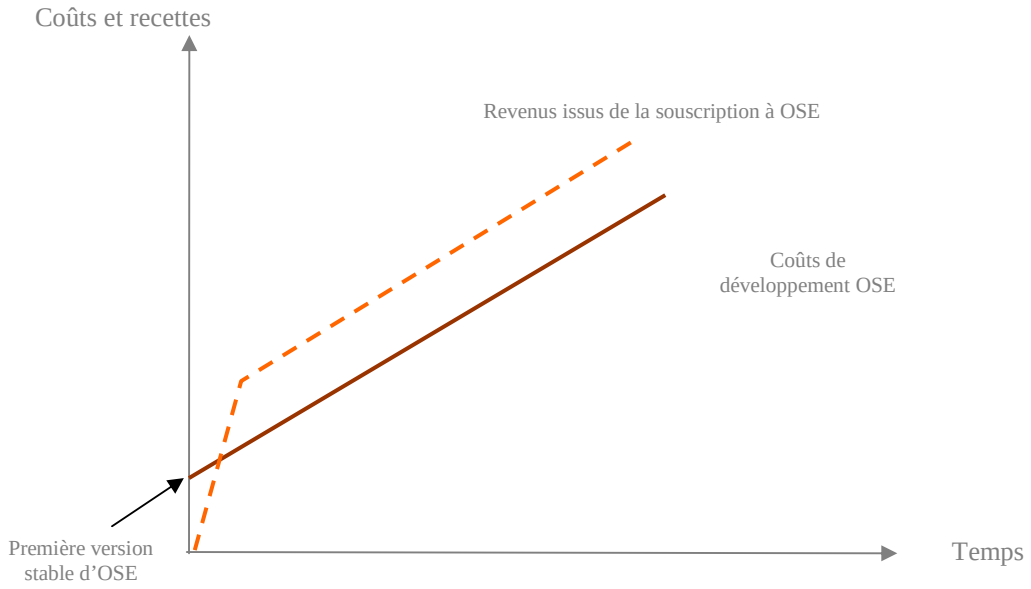
$[S_1 (T_1 ; M_1) ; S_2 (T_2 ; M_2) ; S_3 (T_3 ; M_3) \ldots S_n (T_n ; M_n)]$

Pour ce type de logiciels nous constatons qu'il y a une spécialisation très pointue des firmes dans un domaine précis. La firme créé d'abord un produit incorporant de lourds investissements en recherche et développement : la maîtrise du code source n'est donc pas à la portée de n'importe quel développeur. Ce n'est pas parce qu'un produit est ouvert qu'il permet de mettre tous les acteurs du marché sur un pied d'égalité en matière de concurrence (voir notre travail sur le couplage entre code source et connaissances). L'open source dans ce cas est choisi de manière délibérée. Par exemple, le cas Joram de ScalAgent Technologies est typiquement le type de logiciel entrant dans cette catégorie.

[266] M désigne ici le marché des services associés au produit S_1.

(3b) Dans le domaine du logiciel libre orienté utilisateur final, la firme éditrice essayera de captiver le client en incorporant des fonctionnalités nouvelles (innovations incrémentales) significatives au fur et à mesure. Comme cela est matérialisé dans le graphique ci-dessous.

Figure 41 : Matérialisation des dépenses et recettes d'un logiciel libre[267].



Il n'est pas économiquement viable pour une entreprise d'innover de manière importante puisque le code source du logiciel sera mis à disposition des clients. De plus, il n'est pas possible pour la firme de baser son modèle économique sur une expertise comme un composant.

Comme le souligne l'ex-Directeur général de Mandriva, *« la contrainte de vivre dans l'open source, c'est-à-dire vivre dans un environnement où l'on est à poil tous les matins dans la vitrine. Tous les gens voient ce que l'on fait. Chaque fois que l'on développe un nouveau produit, tout le monde le voit sur Cooker[268]. »*

Par conséquent, la récurrence n'est pas possible comme dans le cas d'un éditeur classique. Le modèle économique typiquement adopté est celui de la souscription.

[267] OSE = produit Open Source d'une Entreprise
[268] *« Cooker »* ici désigne la plateforme publique de développement de Mandriva.

Question de Recherche 5 :

QR 5 : Quels types d'innovations les organisations de l'open source produisent-elles ?

Thèse 5 :

TH 5 : **La majorité des innovations conçues par des utilisateurs dans l'open source sont orientées vers la résolution de problèmes ; lorsqu'elles sont conceptuelles : elles sont concentrés dans trois domaines : la communication, le développement et l'administration de système.**

Question de Recherche 6 :

QR 6 : Quelle différence y a-t-il entre l'innovation dans les domaines du logiciel libre et du logiciel fermé ?

Thèse 6 :

TH 6 : **Les acteurs des modèles libre et fermé ont des visions de l'innovation différentes. D'un côté, les logiciels libres favorisent l'innovation par les extrémités tandis que les logiciels fermés privilégient l'innovation par les concepts.**

A travers nos différentes investigations, nous voyons bien que le modèle d'innovation reposant sur une communauté produisant un logiciel libre a une caractéristique représentant à la fois la force et la limite du modèle. Dans les communautés, il n'y a pas vraiment de réflexion collective pour définir l'objet produit. D'autre part, les acteurs sont impliqués dans des tâches précises dû au découpage du code source en modules sensé diminuer la complexité.

L'idée que nous défendons est que le modèle de la production distribué a une limite importante et que cette limite est intrinsèquement liée à la manière dont les communautés fonctionnent. Il s'agit à la fois de ce qui a fait le succès de l'open source mais c'est aussi une de ses plus grandes limites. Dans l'open source, le processus de développement est dans une logique d'innovation par les extrémités : cela signifie que les acteurs essayent de développer un maximum de fonctionnalités portant sur des améliorations incrémentales remettant le moins possible en question l'architecture globale du logiciel. Il est donc bien rare de voir l'ensemble du logiciel rediscuté par les acteurs.

A l'inverse chez les éditeurs de logiciels on essaye de lancer des produits apportant un élément nouveau sur le marché. Dans ce cas le but n'est pas de répondre à des besoins précis puisqu'il s'agit de les créer. Le raisonnement est donc différent : les communautés recherchent à résoudre des problèmes opérationnels pour cela elles développent ce dont elles ont besoin ; d'un autre côté, les firmes éditrices de logiciels fermés essayent de créer de nouveaux problèmes et de rechercher les solutions permettant de les résoudre (Nonaka 1994: 14).

En somme, c'est la définition même de l'innovation qui diffère entre les communautés de l'open source et les éditeurs du monde fermé. Le but n'étant pas le même les objets produits ne sont pas les mêmes. Même si cette vision est caricaturale de l'open source, il s'agit d'un élément très important dans la logique de raisonnement qui sous-tend ce modèle. Les communautés ne sont donc pas moins innovantes ni plus innovantes : elles développent ce dont elles ont besoin.

Souvent les communautés ont besoin (1) d'alternatives libres n'existant pas sur un système. Elles ont aussi besoin (2) de solutions d'interopérabilité n'existant pas sur le marché et que les firmes dominantes n'ont pas intérêt à produire. Elles développent aussi (3) des logiciels pour lesquels il est très difficile de bâtir un modèle économique du type édition. Ces logiciels ont *per se* une contradiction concernant leur valeur : d'un côté, ils doivent être diffusé au plus grand nombre pour que leur valeur augmente (Arthur 1989) ; d'un autre côté, le fait qu'ils soient vendus ralenti leur diffusion et donc leur valeur. Par conséquent, certains logiciels sont difficilement valorisables sur le marché en tant que tel. C'est le cas des logiciels de communication : il est simple de voir qu'il existe peu de logiciels vendus dans le domaine

de la communication : Skype, Live Messenger, Yahoo! Messenger, etc. sont gratuits. Les logiciels de P2P sont pour la plupart gratuits. Les firmes développant ces logiciels ont bâti un modèle basé sur autre chose que sur l'usage : la publicité. Le modèle publicitaire est une des solutions permettant de détourner ce paradoxe comme nous l'avons vu au travers du cas Mozilla Firefox. Le cas Thunderbird expose le même phénomène mais du point de vue de l'échec de l'application du modèle publicitaire.

PARTIE VI. CONCLUSION : APPORTS, APPLICATIONS ET IMPLICATIONS DES REGIMES DE L'OPEN SOURCE.

Pour conclure, nous synthétiserons les apports d'une thèse sur *les régimes de l'open source* pour une meilleure compréhension des enjeux organisationnels, économiques et technologiques de la diffusion du logiciel libre [1.]. Puis, nous proposerons un modèle d'aide à la décision à destination des gestionnaires de technologies logicielles. Ce modèle permettra aux décideurs de rationnaliser le choix de l'ouverture ou de la fermeture d'une technologie sur la base d'espaces de valeurs identifiés [2.]. Pour finir, nous nous pencherons sur les implications managériales et théoriques de notre recherche notamment en discutant de l'actuel courant dominant de la littérature sur l'innovation : le paradigme de l'open innovation [3.].

1. Les régimes de l'open source : une synthèse des enjeux organisationnels, économiques et technologiques de la diffusion du logiciel libre.

1.1. De l'identification d'un système solidaire et distribué inédit à la modélisation des régimes de l'open source.

Au début de notre recherche, nous avons suggéré une revue des travaux sur les organisations de l'open source. Nous avons orienté notre lecture vers deux types de travaux : les travaux portant sur les communautés d'utilisateurs-développeurs et les recherches traitant de l'implication des firmes.

Dans un second temps, nous avons détaillé les travaux tentant de rattacher le modèle de la communauté d'utilisateurs-développeurs aux formes organisationnelles existantes. Trois cadres conceptuels furent mobilisés dans la littérature : les communautés de pratique et épistémiques, la *« collective invention »* et enfin les réseaux coopératifs. Il était par conséquent nécessaire de réaliser une revue des apports et limites de ces cadres vis-à-vis du phénomène étudié.

Suite à l'étude de ces travaux, nous avons pu constater que l'aspect distribué des communautés open source était relativement bien abordé dans la littérature. En revanche, la littérature était particulièrement lacunaire sur les éléments de solidarité liant les acteurs de

l'open source. De ce fait, nous avons recherché les racines théoriques et pratiques du modèle souche.

A travers notre travail sur les notions de solidarité et de distribution, nous avons réussi à démontrer que la communauté d'utilisateurs-développeurs était issue de la combinaison inédite de deux formes d'organisations : les systèmes de management de la solidarité et les systèmes de production distribués. Plus spécifiquement, nous définissons le modèle racine comme un système de management de la solidarité ouvert où des individus sont liés par une solidarité de conception, d'apprentissage et d'usage. Et d'autre part, cette organisation est également impliquée dans une action collective distribuée où : d'un côté, l'investissement capitalistique est faible du fait de la démocratisation du micro-ordinateur et d'internet ; et d'un autre côté, dont l'objet produit a des propriétés fortement agglutinantes.

Dans un troisième temps, nous avons détaillé les expansions du modèle racine donnant naissance à une variété de nouvelles organisations. Notre travail empirique basé sur cinq cas sélectionnés sur le principe de l'échantillonnage théorique, nous a permis de dériver une taxonomie d'organisations : la firme de l'open source, la communauté d'utilisateurs-développeurs, l'association d'utilisateurs-développeurs, la communauté inter-organisations et le consortium de l'open source.

Nous apportons donc à littérature scientifique à la fois un cadre statique et analytique nommé modèle du pentagone. Ce modèle peut être mobilisé aussi bien pour caractériser les formes organisationnelles existantes, les hybridations et suivre les transformations organisationnelles passées et futures de l'open source.

1.2. La valorisation des logiciels libres : une dynamique nouvelle entre le marchand et le non-marchand.

Notre revue des stratégies et des modèles économiques dans le domaine du logiciel libre, nous a permis d'identifier deux types de modèles d'affaires. Premièrement, les modèles ouverts proposant un déplacement de la valeur du logiciel principalement vers de la prestation de services (expertise, formation, mise à jour, maintenance, etc.). Deuxièmement, les modèles hybrides conjuguant la cession de licence et la prestation de services.

Grâce à nos investigations empiriques, une troisième classe de modèles économiques fut identifiée : il s'agit des modèles fermés. Ces modèles de valorisation sont basés sur des

stratégies de verrouillage reposant sur différents procédés visant à contourner les contraintes imposées par certaines licences libres.

D'autre part, nous découvrons par le biais de différents cas (Firefox, Thunderbird, OpenOffice.org) que les logiciels orientés utilisateur final sont difficilement valorisables avec un modèle ouvert. Seules trois possibilités sont empiriquement identifiées : le financement publicitaire, le modèle de la souscription ou la cession de licence.

En outre, l'étude des stratégies industrielles souligne que les logiciels libres sont utilisés comme des composants génériques sur la base desquels la valeur est créée dans l'industrie. Les composants libres confèrent une indépendance technologique et économique à certains acteurs (intégrateurs) qui jusqu'ici étaient contraints de travailler avec les éditeurs. En revanche, la libération d'un code source ne signifie pas pour autant qu'il y a libre concurrence sur la partie *services associés* aux technologies libérées. La pleine exploitation économique d'une technologie open source nécessite deux catégories de connaissances : l'une est générique (K_1) et peut rapidement être obtenue par un intégrateur, l'autre (K_2) repose sur une expertise donc nécessite un investissement conséquent afin de maîtriser la technologie.

1.3. L'innovation dans l'open source : de l'ouverture de la concurrence à la compensation de l'offre marchande.

La question de l'innovation dans l'industrie du logiciel fait l'objet d'une controverse importante aussi bien du point de vue théorique que du point de vue pratique. Par le biais de nos investigations, nous avons découvert des différences importantes entre les modèles d'innovation ouverts et fermés.

La littérature sur la *« user innovation »* a largement exposé le caractère innovant des utilisateurs dans le processus de conception de produit. Dans le cas de l'open source, ce processus de conception est continu : cela signifie que le logiciel est théoriquement constamment dans une phase de conception.

En revanche, ces travaux se sont rarement penchés sur le type d'innovations générées par les utilisateurs et encore moins sur les communautés d'utilisateurs autonomes. Quelques travaux ont néanmoins proposé d'évaluer l'innovation des logiciels libres. Une étude approfondie de ces recherches a permis d'identifier deux principales limites : d'une part, la confusion entre processus innovant et production innovante ; d'autre part, le manque de critères pertinents pour caractériser l'innovation dans l'industrie du logiciel.

Dans notre travail, nous avons d'abord étudié l'innovation du point de vue du concept général du produit. Grâce à une méthodologie couplant des techniques qualitatives et quantitatives adaptées aux nouvelles possibilités offertes par internet ou la *« web-based Delphi »*, nous avons prouvé empiriquement que la majorité des innovations conçues par des utilisateurs étaient orientées vers la résolution de problèmes. Lorsque ces innovations sont conceptuelles, elles sont concentrées dans trois domaines : la communication, le développement et l'administration de système. En d'autres termes, les communautés d'utilisateurs-développeurs produisent les innovations dont elles ont besoin.

En second lieu, nous nous sommes intéressés aux différences entre les modèles libres et fermés. Après une étude approfondie des enjeux de la conception de logiciels libres et fermés, nous avons constaté que les acteurs de ces modèles avaient des visions très différentes de l'innovation. D'un côté, les acteurs du logiciel libre favorisent l'innovation par les extrémités tandis que les acteurs du logiciel fermé privilégient l'innovation par les concepts. Les visions étant divergentes les innovations produites sont différentes.

2. Les stratégies d'ouverture ou de fermeture des technologies logicielles : de l'arbitrage à la rationalisation.

2.1. Rationaliser les décisions d'ouverture ou de fermeture technologique (1) : la proposition d'un modèle d'aide à la décision.

Il ressort clairement de nos différentes investigations que pour les entreprises le choix de l'open source ou du closed source est principalement motivé par des aspects économiques et technologiques. L'open source n'est donc pas choisi (sauf pour quelques rares exceptions) pour des considérations d'ordre philosophiques surtout pour les grandes entreprises de cette industrie. Nous défendons la thèse que l'open source n'est pas adapté pour la valorisation de tous types de logiciels. Plusieurs exemples peuvent être donnés : SourceForge est passé de l'open source au closed source à cause de difficultés de valorisation (L. Augustin 2002). De même, Roxen eut des difficultés pour bâtir un modèle économique adapté aux logiciels libres de ce fait *« le modèle économique de la compagnie est devenu plus propriétaire »* (Dahlander et Magnusson 2008: 636). De même, le closed source n'est pas adapté à la valorisation économique de tous les logiciels. Les exemples de libérations ne manquent pas : Java, OpenOffice.org, Mozilla, etc.

Suite à nos différentes analyses, nous sommes désormais en mesure de proposer une grille d'aide à la décision à destination des managers de l'industrie du logiciel afin que ces derniers puissent choisir le modèle économique le plus adapté en fonction des espaces de valeurs identifiés pour leurs logiciels.

Tableau 36 : Grille d'identification des espaces de valeur des logiciels.

Espaces de valeur	Modalités		Modèle économique associé	Description
Degré d'innovation de l'usage (EV_1).	Fort (+)	Faible (-)	Vente de licences. Exemples : Adobe (Photoshop), MySQL AB.	Le degré d'innovation de l'usage décrit le niveau d'innovation qui est situé au niveau du concept général du logiciel.
Degré de standardisation de l'usage (EV_2).	Fort (+)	Faible (-)	Développements spécifiques. Exemples : ScalAgent Technologies.	Le degré de standardisation de l'usage correspond à l'importance des modifications nécessaires pour que le logiciel soit utilisable.
Complexité d'utilisation (EV_3).	Fort (+)	Faible (-)	Formation. Exemples : StarXpert.	La complexité d'utilisation correspond à l'apprentissage nécessaire pour utiliser le logiciel.
Niveau de connaissances nécessaire pour la maîtrise de la technologie (EV_4).	Fort (+)	Faible (-)	Expertise technologique. Exemples : ScalAgent (Joram), MobiGIS, eXoPlatform.	Les connaissances nécessaires pour l'appropriation correspondent aux connaissances nécessaires pour que le code source d'un logiciel soit maîtrisé. Lorsque ce niveau est élevé cela signifie que le logiciel apporte une réelle innovation technique.
Importance des rendements d'adoption (EV_5).	Fort (+)	Faible (-)	Publicité, services associés. Exemples : Mozilla Corporation (Firefox), Skype.	Certains logiciels ont intérêt à être distribués car cette diffusion est une partie intégrante de leur valeur. Un logiciel tendra à être diffusé gratuitement en open source si la technologie n'est pas le cœur du modèle économique de la firme ou en closed source si vraiment son modèle économique en dépend car le niveau d'expertise nécessaire (EV_4) n'est pas suffisant pour justifier un modèle économique.
Importance des mises à jour (EV_6).	Fort (+)	Faible (-)	Souscription. Exemples :	Pour certains logiciels les mises à jour sont quasiment indispensables pour l'utilisation du logiciel (exemple : antivirus, logiciels en environnement critique, etc.).
Type de logiciel (EV_7).	Composant (C)	End-user (EU)	Relicensing (Composant).	Le type de logiciel renseigne sur la catégorie de logiciel dont il s'agit.

2.2. Rationaliser les décisions d'ouverture ou de fermeture technologique (2) : cas d'application ex-post et ex-ante.

2.2.1. Cas d'application ex-post.

Notre grille d'aide à la décision peut être mobilisée pour repérer les espaces de valeur liés à un logiciel ceci afin de choisir la stratégie de distribution la plus adaptée aux caractéristiques du logiciel. Il s'agit donc d'un modèle avec une forte contingence[269]. Comme nous l'avons précédemment souligné, ni l'open source ni le closed source ne sont adaptés à la valorisation de tous types de logiciels. Nous développerons quelques cas pour montrer comment cette grille peut être utilisée pour justifier *ex-post* le succès ou l'échec d'un modèle économique et surtout *ex-ante* pour bâtir un modèle économique adapté aux caractéristiques d'un logiciel. Toutefois, cette grille n'est pas une *« recette magique »*, le succès du logiciel dépendra bien évidemment d'autres facteurs. Le but de cette grille est simplement de permettre aux managers de choisir la ou les différents modèles de valorisation adaptés à leurs technologies et d'identifier les espaces de valeurs correspondants.

A. Le cas Firefox : échec de Netscape, succès de la Mozilla Corporation.

Firefox est un logiciel aux caractéristiques suivantes :

$EV_1^{(+)}$ → Navigator (devenu Firefox) était le premier navigateur multi-plateforme lancé sur le marché en 1994 (Jackson, 1999 : 6) et continue d'incorporer de nombreuses innovations d'usage (navigation par onglets, magnétisme des fenêtres[270], réouverture de session, etc.).

$EV_2^{(+)}$ → Le navigateur ne nécessite pas d'adaptations pour pouvoir être utilisé.

$EV_3^{(-)}$ → L'usage d'un navigateur ne nécessite pas de connaissances particulières.

$EV_4^{(+)}$ → Navigator (originellement Mosaic) est le fruit de plusieurs années de recherche au National Center for Supercomputing Applications (NCSA) et d'investissements par Netscape.

$EV_5^{(+)}$ → L'utilisation de Firefox par le plus grand nombre a ouvert la possibilité de vendre de l'espace publicitaire (Google, Ebay, etc.).

$EV_6^{(+)}$ → Les mises à jour sont particulièrement critiques pour l'utilisation du navigateur car les technologies évoluent très rapidement (Java, Ajax, etc.).

$EV_7^{(EU)}$ → Firefox est un produit orienté utilisateur final.

[269] Dans un certain sens notre travail s'inscrit dans la lignée des travaux sur la contingence (Burns et Stalker 1971; Chandler 1989; Lawrence et Lorsch 1989).
[270] Pour une description des deux fonctionnalités citées de Firefox se référer à l'article de Noll (2007).

Pour Firefox, il n'y avait que deux possibilités. (1) Le distribuer sous licence fermé et vendu comme cela a été fait originellement par Netscape. Néanmoins, Firefox aurait fait face à un problème : Navigator et Microsoft Internet Explorer étaient distribués gratuitement et les utilisateurs avaient perdu toute motivation pour payer pour un navigateur internet car il était largement devenu une commodité (Jackson 1999: 74).

(2) Opter pour le modèle publicitaire, c'est ce que la Mozilla Fondation a fait à travers sa filiale (la Mozilla Corporation). C'est certainement la raison qui justifie pourquoi Netscape a échoué avec ce produit car elle n'avait pas d'autres alternatives pour valoriser le logiciel. En effet, la valeur de Firefox réside principalement dans son usage. Firefox est un produit utlisateur-final, Netscape ne pouvait pas se baser sur un modèle de *« dual-licensing »* comme avec un composant.

B. Le cas Photoshop : les raisons de la distribution en closed source payant.

Photoshop est un logiciel aux caractéristiques suivantes :

$EV_1^{(+)}$ → Photoshop est un logiciel dont le concept de base a fait l'objet d'une profonde réflexion.

$EV_2^{(+)}$ → Photoshop ne nécessite pas d'adaptations pour pouvoir être utilisé.

$EV_3^{(+)}$ → L'usage de Photoshop nécessite des connaissances en retouche photo.

$EV_4^{(+)}$ → Photoshop est le résultat de plusieurs années de développement par les frères Knoll, puis par Adobe qui l'a intégré dans ses produits.

$EV_5^{(-)}$ → L'utilisation de Photoshop n'a pas de lien avec la valeur de celui-ci.

$EV_6^{(-)}$ → Le non accès aux mises à jour n'a pas une grande importance dans la mesure où cela n'empêche pas que le produit soit utilisé dans de bonnes conditions.

$EV_7^{(EU)}$ → Photoshop est un produit orienté utilisateur final.

Photoshop d'Adobe est un logiciel distribué sous licence commerciale et il n'est pas prêt de devenir open source. La valeur de Photoshop réside essentiellement dans l'usage de celui-ci. Il y a peu d'espace de valeur ailleurs sauf peut-être si ce n'est dans le domaine de la formation.

C. Le cas Java : l'importance de l'expertise.

Java est un logiciel aux caractéristiques suivantes :

$EV_1^{(-)}$ → Java n'est pas un logiciel situé dans les couches hautes par conséquent il n'introduit pas de nouvel usage.

$EV_2^{(-)}$ → Java a une utilisation très peu standardisée.

$EV_3^{(+)}$ → Java nécessite un apprentissage particulier pour pouvoir maîtriser le langage.

$EV_4^{(+)}$ → Java est un ensemble de technologies très complexes nécessitant une expertise particulière pour pouvoir procéder à des modifications.

$EV_5^{(+)}$ → L'utilisation de Java a une importance particulière dans sa valeur comme tout langage de programmation puisque c'est autant de potentiel de services associés.

$EV_6^{(+)}$ → L'accès aux mises à jour est très important dans la mesure où il s'agit d'un composant sur lequel repose des logiciels développés dans ce langage.

$EV_7^{(C)}$ → Java est un ensemble de technologies de type composant.

La libération de Java n'est pas une surprise si l'on regarde les caractéristiques de cette technologie. C'est un logiciel pour lequel l'expertise est particulièrement importante. La diffusion de Java fait aussi partie intégrante de la valeur de celui-ci, distribuer Java sous licence commerciale n'aurait pas de sens et serait certainement au détriment du logiciel. La distribution la plus adaptée est donc l'open source gratuit. Java est devenu un véritable standard de fait pour l'industrie du logiciel. Par exemple, bon nombre de logiciels développés dans le cadre du Consortium OW2 sont basés sur les technologies Java.

D. OpenOffice.org : les raisons de l'ouverture.

OpenOffice.org est un logiciel aux caractéristiques suivantes :

$EV_1^{(-)}$ → OpenOffice.org n'est pas un logiciel apportant un nouvel usage. La bureautique fait largement parti des commodités.

$EV_2^{(+)}$ → OpenOffice.org fait l'objet d'une utilisation particulièrement standardisée.

$EV_3^{(+)}$ → OpenOffice.org nécessite une phase d'apprentissage afin d'être maîtrisé par l'utilisateur final.

$EV_4^{(+)}$ → OpenOffice.org a une base de code importante (plusieurs millions) il est donc complexe pour des développeurs de maîtriser la technologie. Même si il n'y a pas d'incorporation particulière de recherche et développement dans OpenOffice.org. Il ne s'agit donc pas d'une expertise mais plutôt d'une capacité à pouvoir lire et interpréter le code source. Mais cela est possible car des forks d'OpenOffice.org existent.

$EV_5^{(+)}$ → L'utilisation d'OpenOffice.org a une importance toute particulière car ce logiciel propose le format ODF, un format ouvert de documents visant à réouvrir le marché de la bureautique.

$EV_6^{(-)}$ → L'accès aux mises à jour n'est pas particulièrement important pour l'utilisation du logiciel.

$EV_7^{(EU)}$ → OpenOffice.org est un produit orienté utilisateur final.

La libération d'OpenOffice.org n'est pas surprenante puisqu'il s'agit d'un produit qui n'arrivait pas à dégager suffisamment de valeur en étant commercialisé.

2.2.2. Cas d'application ex-ante.

A. Le cas de deux choix économiques cohérents avec le type de technologie.

Nous allons montrer à l'aide de notre modèle deux exemples fictifs montrant l'adoption d'un modèle de distribution adapté aux caractéristiques des technologies développées : le premier cas porte sur la libération en open source d'un logiciel dont la valeur réside dans du service et le second cas porte sur la vente d'un logiciel dont la valeur réside dans l'usage.

S_1 : $EV_1^{(+)}$; $EV_2^{(+)}$; $EV_3^{(-)}$; $EV_4^{(-)}$; $EV_5^{(-)}$; $EV_6^{(-)}$; $EV_7^{(EU)}$ → Closed source payant.

S_2 : $EV_1^{(-)}$; $EV_2^{(-)}$; $EV_3^{(+)}$; $EV_4^{(+)}$; $EV_5^{(+)}$; $EV_6^{(-)}$; $EV_7^{(C)}$ → Open source gratuit.

A travers ces deux exemples théoriques, nous souhaitons mettre en évidence qu'il n'y a pas de meilleur modèle de distribution *per se*. Tout dépend des caractéristiques technologiques du logiciel que la firme souhaite distribuer. Il convient néanmoins de souligner que distribuer un logiciel dont la valeur réside dans l'usage en open source gratuit est aisé toutefois réaliser le chemin inverse nous semble plus difficile mais possible.

Tandis que la distribution d'un logiciel dont la valeur réside dans des services associés en closed source payant est réversible car la firme peut changer de stratégie et passer à l'open source. La double licence libre et fermée offre la possibilité de revenir en arrière tout en conservant les modifications réalisées par d'autres en demandant aux contributeurs de signer un contrat de partage de copyright.

B. Un cas paradoxal : l'antivirus open source.

Dans nos différentes analyses, nous avons souligné qu'il était complexe de concilier économie de l'usage et open source lorsque l'usage du logiciel était standardisé. Toutefois, il semble qu'il y ait une exception[271] : il s'agit des logiciels dont la mise à jour est indispensable pour le fonctionnement du logiciel. Nous pouvons donc tout à fait imaginer le cas d'une entreprise proposant une licence d'utilisation pour un antivirus distribué sous licence libre

[271] Une proposition est scientifique si elle est falsifiable, c'est ce que nous montrons avec cette exception.

puisque l'utilité d'un antivirus réside justement dans le fait que sa base de données virale soit à jour.

3. Implications théoriques et managériales des régimes de l'open source : vers une contingence des stratégies ouvertes et fermées ?

Notre thèse nous a permis de présenter les principaux enjeux de l'open source dont nous avons détaillé les régimes. Nous avons pu constater que la gestion d'un logiciel faisait appel à au moins trois domaines interdépendants : l'organisation, la technologie et la valorisation. En effet, avant de pouvoir valoriser une technologie il faut au préalable la concevoir ce qui signifie donc organiser le développement de celle-ci. L'organisation, la technologie et la valorisation sont étroitement liés pour l'ensemble des organisations de l'industrie du logiciel.

Les résultats de notre thèse ont deux importantes implications théoriques et managériales : les stratégies ouvertes et fermées sont complémentaires car ni l'une ni l'autre ne sont adaptées à la valorisation de toutes les technologies [3.1.] ; les activités de recherche et développement (R&D) sont au cœur de la construction de l'avantage concurrentiel des firmes de l'open source bien que ce rôle soit actuellement débattu dans la littérature [3.2.].

3.1. Les stratégies d'ouverture et de fermeture : de l'opposition à la complémentarité.

Actuellement, le paradigme de l'open innovation (Chesbrough 2003) devient de plus en plus dominant dans la littérature sur l'innovation : de nombreux travaux s'insèrent dans ce paradigme (Dahlander et Wallin 2006; Gruber et Henkel 2006; Henkel 2006; Rossi, Bonaccorsi, et Monica 2007) pour n'en citer que quelques-uns. Ce paradigme défend l'idée que le partage de l'innovation *« Open Innovation Paradigm »* est devenu un impératif car celui-ci renforce la performance économique des firmes pratiquant ces politiques. A l'inverse, ce paradigme prétend que la rétention des innovations *« Closed Innovation Paradigm »* a un effet néfaste sur la performance des firmes pratiquant cette stratégie du fait de son inadaptation au contexte économique et technologique actuel.

Dans notre thèse, nous avons montré que la divulgation des innovations n'était pas toujours compatible avec certains logiciels dont la valeur résidait principalement dans l'usage. En d'autres termes, nous avons prouvé que l'open source n'était pas adapté pour valoriser

l'ensemble des logiciels : les difficultés des firmes à construire des modèles économiques avec ce type de logiciels est un des arguments les plus probants soutenant cette thèse (Sharma et al. 2002: 7). De même, le retour à des stratégies fermées par un certain nombre de firmes comme Roxen (Dahlander et Magnusson 2008: 636) ou la firme distribuant SourceForge (L. Augustin 2002) soutient les limites des stratégies d'open innovation. D'un autre côté, il n'est pas non plus possible de valoriser tous types de logiciel avec des stratégies fermées. La coexistence des économies closed et open source est l'argument le plus convainquant en faveur de cette thèse. Le cas présentant la stratégie open source de Thales et celui du consortium OW2 ont démontré que les logiciels libres pouvaient être utilisés comme des composants permettant de développer des solutions *sur-mesure* en embarquant à la fois des modules libres et fermés.

3.2. Le rôle des activités de R&D dans la construction de l'avantage concurrentiel des firmes de l'open source.

Les activités de R&D interne (dans le cadre de logiciels open source de type composants) sont essentielles pour la création d'un avantage concurrentiel. En effet, c'est sur ces connaissances que ces firmes basent leur modèle économique. Pour une firme développant un composant libre celle-ci doit d'une part réussir à inciter les intégrateurs à utiliser ses technologies mais également garder un contrôle technologique sur une partie stratégique du logiciel pour pouvoir bâtir de la valeur sur une expertise associée à cette technologie. Il est donc impératif pour une firme de ne pas partager trop tôt sa technologie de manière à avoir une avancée conséquente en matière de connaissances développées au cours des premières phases de conception (R&D).

Les entreprises ayant réussi à développer une réelle activité d'édition associée à un logiciel libre ont pour la plupart investit de manière importante en recherche et développement (R&D) comme par exemple ScalAgent Distributed Technologies avec Joram ou Thales avec son système de gestion du trafic aérien (Thalix). Sans cet investissement en R&D, une firme ne peut pas développer suffisamment de connaissances de niveau 2 (K_2) et risque d'être concurrencée par une autre firme ou une organisation réalisant un *fork.*

Dans son ouvrage, Chesbrough parle du problème du syndrome du *« not invented here »* (Chesbrough, 2003a : 30) alors qu'aujourd'hui on a exactement un inversement de tendance dans la littérature que nous serions tentés de nommer syndrome du *« not invented elsewhere »*

consistant à croire que toutes les innovations proviennent de l'extérieur. Il s'agit d'un véritable problème car la littérature semble considérer que la R&D interne n'est plus capable d'innover. Pourtant Chesbrough lui-même explique que dans certaines industries la fonction R&D interne reste adaptée pour gérer l'innovation. La propriété intellectuelle est très forte dans ces industries (Chesbrough 2003: 34).

Les travaux de Chesbrough ont une limite certaine concernant le type d'industrie étudié puisque l'auteur base ses analyses sur une industrie particulièrement spécifique : l'industrie informatique. Cette industrie est caractérisée par plusieurs phénomènes particuliers qu'il n'est pas envisageable de généraliser à toutes les industries : elle est caractérisée par une structure en *« plateformes »* et de plus en plus orientée vers la notion de services.

(1) L'industrie du matériel est structurée en plate-forme. Tous les fabricants de matériels doivent respecter les normes et les standards définis de manière plus ou moins concertée. De même, l'industrie du logiciel tend à se structurer dans cette même logique. Il existe aussi des plateformes dans le domaine du logiciel comme Windows ou les composants middleware développés par le consortium OW2 (Thomas 2009). Les développeurs de logiciels doivent respecter les normes édictées par Microsoft pour son système d'exploitation. Dans ce contexte de structuration de produit où nous avons une véritable *économie en modules*, l'open innovation est certainement le plus pertinent paradigme mais pas pour toutes les technologies.

(2) Le deuxième phénomène est que l'industrie informatique est de plus en plus orientée vers la notion de services que se soit au niveau matériel ou logiciel. Pour le matériel par exemple : des entreprises se sont spécialisées dans la mise à disposition de serveurs plutôt que la vente de serveurs. Pour les logiciels, des entreprises proposent des services en ligne. Google est l'exemple le plus caricatural d'une entreprise ayant bâti un modèle économique uniquement basé sur des services en ligne. L'open innovation va également dans le sens d'une économie de services.

En revanche, dans les industries basées fortement sur l'usage et non les services, les politiques d'ouverture semblent moins adaptées. Typiquement dans le domaine pharmaceutique, il est évident qu'un laboratoire ne partagera pas la nouvelle molécule qu'il vient d'élaborer avec ses concurrents et partenaires car il est quasiment l'unique acteur de la chaîne de valeur. Il en est de même dans le domaine des semences génétiquement modifiées.

En somme, notre thèse contribue à la meilleure compréhension des logiques respectives des stratégies ouvertes et fermées en mettant en exergue leurs avantages et limites. Nous invitons donc les chercheurs académiques à ouvrir le débat autour de ce sujet dans d'autres industries.

Table des illustrations.

Les contributeurs à cette thèse sont :

M. Bernard (Abul), I. Chris (Accuride Canada Inc), S. Will (Album Shaper), F. Roy (Apache Software Foundation), R. Martin (Areva), V. Gilles (Asbench), A. Alexandre (Aspec), C. Marcelo (Aspec), S. Romero (Aspectrum), P. Michael (Audiocutter), M. Chris (Autohotkey), Z. Andres (Automkv), Johnny (Avijoin), N. Alexander (Avi-Mux), D. Pierre (Avisynth), S. Randolf (Ayam), P. Nigel (Basilisk Ii), F. Jamie (Bersirc), C. Nick (Bersirc), C. David (Blueberry), S. Nick (Camstudio), D. Ulrich (Caprice32), Albert (Cdex), S. Jörg (Cdrecord), V. Oliver (Cdrtfe), D. Andreas (Cedocida), Z. Sofia (CNRS), M. Yiannis (Codeblocks), K. Marc (Codebrowser), G. Jens (Crimson Fields), T. Jorrit (Crystal Space), S. Eric (Crystal Space), P. Derek (CVS), P. Alexandre (Cvsgui), A. Jean-Paul (Dalibo), H. Tobias (Danish Navy), J. David (Dave Gnukem), P. Todd (Dc++), H. Mika (Delirium), S. David (Delphamp), L. Colin (Dev C++), M. Steffen (Dia), B. Tim (Dirac), V. Bjarke (Directui), V. Philippe (Dokeos), Peter (Dosbox), B. Tomas (Dvd2Avi), H. Thorsten (Dz Bank), Z. Jan (Eclipse), B. Ben (Egoboo), S. Chris (Emerge Desktop), T. Sami (Eraser), Suprie (Ericsson), H. Ben (Exoengine), A. Philippe (Exoplatform), N. John (Explore2Fs), W. Matt (Ext2Fsd), B. Fabrice (Ffmpeg), J. Lee (Ffs Driver), K. Tim (Filezilla), A. Attila (Filmshrink), C. Olson (Flightgear), V. Jeroen (Fox-Toolkit), V. Albert (Freebasic), R. Juergen (Freecad), H. Chen (Freedos), H. Arthur (Freedroidrpg), C. Carl (Freepascal), A. Paul (Freeworks), C. Jeff (Freshmeat), Daniel (Freshmeat), H. Torben (Galan), H. Ales (Geda), B. Joel (Geoshell), L. Russel (Ghostscript), J. Ray (Ghostscript), U. Andreas (Gltron), D. Chris (Google), Wef (Gordian Knot), D. Laurent (Graoumf Tracker 2), A. Cédric (Grisbi), T. John (Gxexplorer), C. Lee (Hi/Fn 7751 Driver), C. Alexandro (Hydrogen), C. Jason (I2P), K. Christian (Infra Recorder), H. Bryce (Inkscape), K. Emilien (INRIA), R. Fy (INRIA), C. Mathieu (INRIA), C. Pierre-Emmanuel (INRIA), Q. Samuel (INRIA), M. Nicolas (INRIA), G. Nikolaus (Irrlicht 3D), C. Alan (Issendis), William (K2Wrpg), S. Cort (Kafka), H. Marc (Kcdread), P. Cédric (Kexi), S. Sebastian (Kexi), S. Jaroslaw (Kexi), P. Tomasz (Kgb Archver), R. Glenn (Libpng), V. Daniel (Libxml2), S. Grégory (Lincity), M. Rob (Liteshell), D. Jean (Lm-Sensor), E. Philip (Lm-Sensor), W. Galazka (Long File Name Services), L. Sam (Maelstrom), D. Loïc (Mekensleep), K. Ruediger (Micq), K. Shy (Musepack), R. Guillaume (Muzip), G. Ethan (Nagios), D. Renaud (Nessus), R. Pat (Nethack), Fyodor (Nmap), B. Florian (Notepad2), F. Alessio (Ocb), M. Dan (Ogm Tools), V. Ludovic (Ogm Tools), S. Steve (Ogre 3D), S. Yaya (Omg), W. Philip (Openal), Upi (Openmortal), H. David (Openmsx), B. Cédric (Openoffice.Org), K. Marie-Jo (Openoffice.Org), G. Tony (Openoffice.Org), G. Laurent (Openoffice.Org), S. Van (Opensourcemark), S. Jean-Noël (Orthophile), r0lZ (Pgcedit), Y. Nate (Pgina), G. Albrecht(Phproject), S. Loïc (Pidgin), E. Sean (Pidgin), B. Ross (Portaudio), H. Art (Pythoncad), K. Todd (Radeontweaker), W. Daniel (Regcompact), J. Vincent (Request Tracker), K. Gerson (Rfstool), R. Dominique (Rili), B. Sylvain (Savannah), S. Franz (Scribus), V. Erwan (Seanodes), Jules (Serenade), I. Courtland, (Serenade), W. David (Simple Directory Listing), M. Joel (Slab3D), R. Noa (Smart Package), W. Tom (Snackamp), S. Anthony (Stella), M.Bradford (Stella), L. Rene (Subtitle Creator), V. Erik (Subtitle Creator), T. Erwin (Sun Microsystems), M. James (Supercollider), A. Alexander (Taskswitchxp), L. Laurent (Thales), K. Lajos (Tridcomm), D. Philippe (Ubikis), M. Nehal (Ufstools), S. Franc (Unedic), Alberto (University Of Bergamo), C. Antoine (Videolan), Typz (Virtual Dimension), D. Felipe Montero (Virtual Magnyfying Glass), P. John (Virtuawin), V. Jamie (Webgui), C. Jamie (Webmin), S. Oliver (Windirstrat), S. Bernard (Windirstrat), Worfje (Winfellow), G. Bjorn (Wings 3D), G. Dan (Wings 3D), W. Toni (Winuae), B. Ben (Withheld), T. Marton (Xaos), K. Zoltan (Xaos), O. Jan (Xaos), B. Hanno (Xfdisk), N. Derek (Xpdf), C. Alejandro (Yafray), W. Mathias (Yafray), W. Michael, A. Fabricio, F. John, C. Daniel, G. Kim, Jared, H. Patrick, Calculus, Bjorn Even, Frank, Anders, H. Fathir, Gunner, Tomaz, Richard, Thierry, Eivind, Arjun, Ray, Wilson, Manish, Bill, Shawn, Michael, Robert, Kristensen, Bjorn, Wahyu, Martin, Iztok, D. Neil, Troy, Egor, Christopher, Imran, Rob, Fred, Robert, Ashish, Josh, D. Jan, D. Dermot, F. Brandon, S. Wipat, R. Simón, S. Fredrik, T. Anders, H. Jensen, E. Stian Rodven, F. Rikard, Kyle, Noem, Nate, Jakob, David, Drukenbatman.

Ainsi que tous les oubliés…

Bibliographie

Acquier, Aurélien (2007), 'Les modèles de pilotage de développement durable : du contrôle externe à la conception innovante (Thèse de doctorat en sciences de gestion)', (Ecole des Mines de Paris).

Afriat, Yoan 'Iliad (free) assignée par des développeurs de logiciels libres pour violation de la licence GNU/GPL', <http://junon.u-3mrs.fr/u3ired01/Main%20docu/PLA/24nov08-iliad.htm>.

Aggeri, Frank (2008), 'Régénérer les cadres de la stratégie : Mise en dispositif et exploration de nouveaux espaces d'action stratégiques, Dossier d'Habilitation à Diriger des Recherches (HDR)', (Université Paris-Dauphine).

Aigrain, Philippe (1980), 'De l'Organisation au Calcul… Le temps de Charles Babbage'. <http://paigrain.debatpublic.net/docs/babbage.pdf>.

Alexy, Oliver et Henkel, Joachim (2007), 'Promoting the Penguin: Who is Advocating Open Source Software in Commercial Settings?', *European Academy of Management (EURAM)* (HEC Paris, Jouy-en-Josas).

Allen, Robert C. (1983), 'Collective Invention', *Journal of Economic Behavior and Organization*, (4), 1-24.

AlMarzouq, Mohammad, Zheng, Li, Rong, Guang, et Grover, Varun (2005), 'Open Source: Concepts, Benefits, And Challenges', *Communications of the Association for Information Systems* (16), 756-84.

Apple Computer Inc. 'AppleMasters', <http://web.archive.org/web/20000610210741/www.apple.com/applemasters/jknoll/index.html>.

Ardener, Shirley (1964), 'The Comparative Study of Rotating Credit Associations', *The Journal of the Royal Anthropological Institute of Great Britain and Ireland*, 94 (2), 201-29.

Arthur, Brian W. (1989), 'Competing Technologies, Increasing Returns, And Lock-In By Historical Events', *The Economic Journal*, 99 (March), 116-31.

Ashley, William James (1900), *Histoire et doctrines économiques de l'Angleterre*, ed. Giard V. & Brière E., trans. Bondoit P. & Bouyssy Savinien 259-85.

Aubert, Francis et Gaigné, Carl (2005), 'Histoire de la dynamique territoriale de l'industrie - Le rôle de la demande de travail', *Cahiers d'économie et sociologie rurales*, 76, 49-70.

Audiganne, Armand (1860), *Les populations ouvrières et les industries de la France : études comparatives sur le régime et les ressources des différentes industries* (1: Capelle) 91-111.

Augustin, L. 'A message from VA Software Corporation CEO Larry Augustin.', <http://sourceforge.net/forum/forum.php?forum_id=164916>.

Augustin, Thierry (1840), *Considérations sur l'histoire de France* (J. Tessier).

Auray, Nicolas (2003), 'Le sens du juste dans un noyau d'experts Debian et le puritanisme civique', in Serge Proulx, Françoise Massit-Folléa, et Bernard Conein (eds.), *Internet. Une utopie limitée. Nouvelles régulations, nouvelles solidarités* (Éthique et philosophie de la communication; Montréal: Presses Universitaires de l'Université du Québec).

Auray, Nicolas (2004), 'La régulation de la connaissance : arbitrage sur la taille et gestion aux frontières dans la communauté Debian', *Revue d'économie politique*, 160-82.

Auray, Nicolas (2006), 'Les configurations de marché du logiciel et le renouvellement du capitalisme', in François Eymard-Duvernay (ed.), *L'économie des conventions, méthodes et résultats (Tome 2)* (La Découverte).

Auriemma, Luigi 'Anytoiso GPL Violation', <http://aluigi.altervista.org/misc/anytoiso_gpl_violation.txt>.

Auvray, L. (1862), 'Monographie n°24 : La lingère de Lille', in Frédéric Le Play (ed.), *Les Ouvriers des deux mondes* (3: Société internationale des études pratiques d'économie sociale), 247-84.

Axelrod, Robert (1992), *Donnant Donnant - Théorie du comportement coopératif*, trans. Michèle Garène (Collection Sciences Humaines: Editions Odile Jacob).

Babbage, Charles (1832), 'On the economy of machinery and manufactures'.

Baldwin, Carliss Y. et Clark, Kim B. (2005), 'Between "Knowledge" and "the Economy": Notes on the Scientific Study of Designs', *Scientific Studies of Designs* (Boston: Harvard Business School), 1-37.

Baldwin, Carliss Y. et Clark, Kim B. (2006), 'The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model?', *Management Science*, 52 (7), 1116-27.

Bancilhon, François (2007), 'Logiciel libre et gratuité', *Don et gratuité, Science & Devenir de l'Homme, Les Cahiers du M.U.R.S* (Paris, Collège de France), 77-88.

Baudet-Dulary, Alexandre-François (1832), 'Considérations sur le Phalanstère, La Réforme industrielle ou le Phalanstère,'.

Benkeltoum, Nordine (2006), 'Les organisations de l'industrie du logiciel libre', *Unpublished MA Dissertation* (Paris: Université Paris X - Ecole des Mines de Paris).

Benkeltoum, Nordine (2007), 'Open Source Culture as New Management Knowledge: Towards New Cycles of Innovative Business', *European Academy of Management (EURAM)* (HEC Paris, Jouy-en-Josas).

Benkeltoum, Nordine (2008a), 'Les régimes de l'open source : Management, Innovation, Solidarité et Business Models', *Cercle Doctoral Européen de Gestion (CDEG), FNEGE* (Beaulieu, France).

Benkeltoum, Nordine (2008b), 'Open source regimes: Management, Innovation, Solidarity and Business Models', *Doctoral Workshop of the 15th International Product Development Management (IPDM) Conference* (Hamburg, Germany).

Benkeltoum, Nordine et Hatchuel, Armand (2008), 'Are open source software communities a source of disruptive innovation?', in EIASM (ed.), *Proceedings of the 15th International Product Development Management (IPDM) Conference* (Hamburg, Germany).

Benkler, Yochai (2002), 'Coase's Penguin, or Linux and the Nature of the Firm', *Yale Law Journal*, 112 (3), 369-446.

Benkler, Yochai (2006), *The Wealth of Networks: How Social Production Transforms Markets and Freedom* (New Haven: Yale University Press).

Benkler, Yochai et Nissenbaum, Helen (2006), 'Commons-Based Peer Production and Virtue', *Journal of Political Philosophy*, 14 (4), 394-419.

Beziat, Gérard (1992), 'Relever les challenges liés aux fusions et acquisitions', in Claude Lemoine (ed.), *Evaluation et Innovation dans les Organisations* (Sciences Humaines: EAP, Sciences Humaines), 101-10.

Bitzer, Jürgen et Schröder, Philip J. H. (2005), 'Bug-fixing and code-writing: the private provision of open source software', *Information Economics and Policy*, 17 (3), 389-406.

Boa, Treize 'ScummVM dans des jeux Atari, au mépris de la GPL', <http://linuxfr.org/2009/06/25/25651.html>.

Boiteux, Nicolas 'L'ésotérisme du bazar, et la pomme empoisonnée démystifiée', <http://membres.lycos.fr/code34/esoterisme_du_bazar.html>.

Bonaccorsi, Andrea et Rossi, Cristina (2003a), 'Why Open Source software can succeed', *Research Policy*, (32), 1243-58.

Bonaccorsi, Andrea et Rossi, Cristina (2003b), 'Comparing motivations of individual programmers and firms to take part in the open source movement - From community to business', (Pisa: Laboratory of Economics and Management, Sant'Anna School of Advanced Studies).

Bonaccorsi, Andrea, Giannangeli, Silvia, et Rossi, Cristina (2006), 'Entry Strategies Under Competing Standards: Hybrid Business Models in the Open Source Software Industry', *Management Science*, 52 (7), 1085-98.

Bourgeois, Léon (1998), *Solidarité* (Villeneuve d'Ascq: Presse Universitaires du Septentrion).

Brown, John Seely et Duguid, Paul (1991), 'Organizational learning and communities-of-practice: toward a unified view of working, learning and innovation', *Organization Science*, 2 (1), 40-57.

Buchanan, James M. (1965), 'An Economic Theory of Clubs', *Economica, New Series*, 32 (125), 1-14.

Burns, Thomas et Stalker, George Macpherson (1971), *The Management of innovation* (2nd Edition edn., Social Science Paperback; London: Tavistock Publications).

Callon, Michel, Cohendet, Patrick, Curien, Nicolas , Dalle, Jean-Michel, Eymard-Duvernay, François, Foray, Dominique, et Schenk, Eric (1999), *Réseau et Coordination* (Paris: Economica).

Capobianco, Fabrizio 'The Honest Public License', <http://www.funambol.com/blog/capo/2006/08/honest-public-license.html>.

Chanal, Valérie (2000), 'Communautés de pratique et management par projet : A propos de l'ouvrage de Wenger (1998) Communities of Practice: Learning, Meaning and Identity', *M@n@gement*, 3 (1), 1-30.

Chanal, Valérie et Mothe, Caroline (2005), 'Concilier innovations d'exploitation et d'exploration : le cas du secteur automobile', *Revue Française de Gestion*, 31 (154), 173-91.

Chandler, Alfred D. Jr. (1989), *Stratégies et structures de l'entreprise* (Paris: Editions d'Organisations).

Chandy, Rajesh K. et Tellis, Gerard J. (1998), 'Organizing for Radical Product Innovation: The Overlooked Role of Willingness to Cannibalize', *Journal of Marketing Research*, 35 (November), 474-87.

Chesbrough, Henry (2003), *Open Innovation: The New Imperative for Creating and Profiting from Technology* (Harvard Business School Press).

Chevalier, Jean et Considérant, Victor (1832), 'Introduction au prospectus général', *Journal le Phalanstère ou la Réforme Industrielle*, (1).

Christensen, Clayton M. (1997), *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail* (The Management of Innovation and Change Series; Boston, Massachusetts: Harvard Business School Press).

Christensen, Clayton M. (2006), 'The Ongoing Process of Building a Theory of Disruption', *Journal of Product Innovation Management*, 23, 39-55.

Christensen, Clayton M. et Overdorf, Michael (2000), 'Meeting the Challenge of Disruptive Change', *Harvard Business Review*, March–April.

Christensen, Clayton M. et Raynor, Michael E. (2003), *The Innovator's Solution: Creating and sustaining successful growth* (Boston, Massachusetts: Harvard Business School Press).

CNRS (2004a), 'Tontine', *Trésor de la Langue Française Informatisé*.

CNRS (2004b), 'Agglutiner', *Trésor de la Langue Française Informatisé*.

CNRS (2004c), 'Distribuer', *Trésor de la Langue Française Informatisé*.

CNRS (2004d), 'Ghilde', *Trésor de la Langue Française Informatisé*.
CNRS (2004e), 'Souscription', *Trésor de la Langue Française Informatisé*.
Coase, Ronald Harry (1937), 'The Nature of the Firm', *Economica*, 4 (16), 386-405.
Cohendet, Patrick, Créplet, Frédéric, et Dupouët, Olivier (2003), 'Innovation organisationnelle, communautés de pratique et communautés épistémiques : le cas de Linux', *Revue Française de Gestion*, 29 (146), 99-121.
Considérant, Victor (1832), 'Lettre au maréchal Soult sur l'entreprise du Phalanstère, et demande d'un congé', *La Réforme industrielle ou le Phalanstère*, 84-86.
Cour de Justice des Communautés européennes 'Pronuptia de Paris GmbH contre Pronuptia de Paris Irmgard Schillgallis', (Arrêt de la CJCE du 28 janvier 1986).
Couveignes, Jean-Marc (2009), 'Un exemple de code correcteur', (Université de Toulouse).
Crémer, Jacques et Gaudeul, Alexandre (2004), 'Quelques éléments d'économie du logiciel libre', *Réseaux*, 22 (124), 111-39.
Crozier, Michel et Friedberg, Erhard (1977), *L'acteur et le système* (Editions Le Seuil).
Cusumano, Michael A. (1991), *Japan's Software Factories* (New York: Oxford University Press).
Cusumano, Michael A. (2004), 'Reflexion on Free and Open Source Software', *Communications of the ACM*, 47 (10), 25-27.
Dahlander, Linus (2004), 'Appropriating the Commons: Firms in Open Source Software', (Chalmers University of Technology), 1-23
Dahlander, Linus et Magnusson, Mats (2005), 'Relationships between open source software companies and communities: Observations from Nordic firms', *Research Policy*, 34 (4), 481-93.
Dahlander, Linus et McKelvey, Maureen (2005), 'Who is not developing open source software? non-users, users, and developers', *Economics of Innovation and New Technology*, 14 (7), 617-35.
Dahlander, Linus et Wallin, Martin W. (2006), 'A man on the inside: Unlocking Communities as complementary assets', *Research Policy*, 35 (7), 1243-59.
Dahlander, Linus et Magnusson, Mats (2008), 'How do Firms Make Use of Open Source Communities?', *Long Range Planning*, (41), 629-49.
Dahlin, Kristina B. et Behrens, Dean M. (2005), 'When is an invention really radical? Defining and measuring technological radicalness', *Research Policy*, (34), 717-37.
Dalle, Jean-Michel, David, Paul A., Ghosh, Rishab A., et Steinmueller, Edward W. (2005), 'Advancing Economic Research on the Free and Open-Source Software Mode of Production', in Marleen Wynants et Jan Cornelis (eds.), *How Open is the Future? Economic, Social & Cultural Scenarios inspired by Free & Open-Source Software* (Brussels: VUB Brussels University Press), 395-429.
Danneels, Erwin (2004), 'Disruptive Technology Reconsidered: A Critique and Research Agenda', *Journal of Product Innovation Management*, 21, 246-58.
Danneels, Erwin (2006), 'From the Guest Editor Dialogue on the Effects of Disruptive Technology on Firms and Industries', *Journal of Product Innovation Management*, 23, 2-4.
Darche, Marc-Aurèle, Séguin, Laurent, Fermigier, Stefane, Elie, François, Di Cosmo, Roberto, et Meister, Guillaume 'Modèles économiques liés aux logiciels libres', <http://www.aful.org/professionnels/modeles-economiques-logiciels-libres>.
David, Albert (2000), 'Logique, épistémologie et méthodologie en sciences de gestion : trois hypothèses revisitées.', in Albert David, Armand Hatchuel, et Romain Laufer (eds.), *Les nouvelles fondations des sciences de gestion, éléments d'épistémologie de la recherche en management* (Paris: Vuibert).

David, Albert (2008), 'Recherche et intervention', *Cercle Doctoral Européen de Gestion (CDEG), FNEGE* (Beaulieu, France).

David, Albert et Hatchuel, Armand (2007), 'From Actionable Knowledge to Universal Theory in Management Research', in Rami A.B. Shani, Susan Albers, William A. Pasmore, Bengt Stymne, et Niclas Adler (eds.), *Handbook of Collaborative Management Research* (SAGE).

De Hoefer, Jean Chrétien Ferdinand (1858), *Nouvelle biographie générale depuis les temps les plus reculés jusqu'à de nos jours, avec des renseignements bibliographiques et l'indication des sources à consulter* (27; Paris: Firmin Didot Frères).

de Prony, Gaspard Riche (1824), 'Notice sur les grandes tables logarithmiques et trigonométriques adaptées au nouveau système métrique et décimal', (Paris: Imprimerie de Firmin Didot).

de Vaujany, François-Xavier (2007), 'La relation pratiques religieuses-pratiques managériales : une approche historique ', *XVIème Conférence Internationale de Management Stratégique* (AIMS), 1-26.

Deek, Fadi P. et McHugh, James A. (2007), *Open Source: Technology and Policy* (New York: Cambridge University Press).

Demazière, Didier, Horn, François, et Zune, Marc (2007), 'The Functioning of a Free Software Community: Entanglement of Three Regulation Modes Control, Autonomous and Distributed ', *Science Studies,* 20 (2), 34-54.

Demazière, Didier, Horn, François, et Zune, Marc (Forthcoming), 'La dynamique de développement des « communautés » du logiciel libre: conditions d'émergence et régulations des tensions', *Revue Terminal*.

Demil, Benoit et Lecocq, Xavier (2005), 'Neither market nor hierarchy or network: the emerging Bazaar Governance', (Lille: Université Lille 3), 1-25.

Demsetz, Harold (1967), 'Towards a Theory of Property Rights', *The American Economic Review,* 57 (2), 347-59.

Department of Justice (1998), 'Justice Department Files Antitrust Suit Against Microsoft For Unlawfully Monopolizing Computer Software Markets.'.

Dewar, Robert D et Dutton, Jane E. (1986), 'The Adoption of Radical and Incremental Innovations: an Empirical Analysis', *Management Science,* 32 (11), 1422-33.

Di Benedetto, Anthony C. (2006), 'From the Editor. Welcome to a new volume of JPIM!', *Journal of Product Innovation Management*, 23, 1.

Diderot, Denis et d'Alembert, Jean Le Rond (1751-1782), 'Encyclopédie ou Dictionnaire raisonné des sciences, des arts et des métiers, par une Société de Gens de lettres'.

Drukenbatman 'The dirge of NeoOffice', <http://www.drunkenblog.com/drunkenblog-archives/000706.html>.

Duguid, Paul (2006), 'Un retour sur le travail de Julian Orr', (23 novembre 2006, Séminaire du Centre de Recherche en Gestion (CRG), Ecole Polytechnique.).

Dumez, Hervé (2007), 'Situation de travail, apprentissage et organisation : retour sur le travail de Julian Orr", notes du séminaire de Paul Duguid, 23 novembre 2006', *Le Libellio d'Aegis,* 3 (1), 13-16

Durkheim, Emile (1889), 'Communauté et société selon Tonnies', *Revue Philosophique,* (27), 416-22.

Duval, Gael 'Ma propre FAQ sur Mandriva Linux', <http://www.indidea.org/gael/fr/gael-answers.php>.

Dzimira, Sylvain (2006), 'Une vision du paradigme du don : Don, juste milieu et prudence', 1-13.

Edwards, Kasper (2005), 'An economic perspective on software licenses—open source, maintainers and user-developers', *Telematics and Informatics,* (22), 111-33.

Eisenhardt, Kathleen M. (1989), 'Building Theories from Case Study Research', *The Academy of Management Review*, 14 (4), 532-50.
Eisenhardt, Kathleen M. et Graebner, Melissa E. (2007), 'Theory building from cases: opportunities and challenges', *Academy of Management Journal*, 50 (25-32).
Elliott, Margaret S. et Scacchi, Walt (2003), 'Free Software: A Case Study of Software Development in a Virtual Organizational Culture', (University of California), 1-73.
Fielding, Roy T. (1999), 'Shared Leadership in the Apache Project', *Communications of the ACM*, 42 (4), 42-43.
FIguiere, Hubert (2007), 'Office Open XML: a technical approach for OOo', *OOoCon 2007* (September 21st; Barcelona).
FingerPrint 'Email client market share', <http://fingerprintapp.com/email-client-stats>.
Fitzgerald, Brian (2007), 'The Transformation of Open Source Software', *MIS Quarterly*, 30 (3), 587-98.
Fourier, Charles (1822a), *Oeuvres complètes de Charles Fourier, Tome deuxième, Théorie de l'unité universelle, premier volume, deuxième édition* (Paris: Société pour la propagation et la réalisation de la théorie de Fourier).
Fourier, Charles (1822b), *Traité de l'association domestique-agricole* (1; Paris: Bossange Père et Mongie P. Ainé).
Fourier, Charles (1832), 'Programme de la fondation proposée', *Journal le Phalanstère ou la Réforme Industrielle,* (1), 7-12.
Fourier, Charles (1845a), *Le nouveau monde industriel et sociétaire (sections I, II et III)* (Tremblay, Jean-Marie edn., Les classiques des sciences sociales; Chicoutimi: Cégep, Université du Québec).
Fourier, Charles (1845b), *Le nouveau monde industriel et sociétaire (sections IV, V, VI et VII. Postface)* (Tremblay, Jean-Marie edn., Les classiques des sciences sociales; Chicoutimi: Cégep, Université du Québec).
Franke, Nikolaus et von Hippel, Eric (2003a), 'Satisfying heterogeneous user needs via innovation toolkits: the case of Apache security software', *Research Policy*, 32 (7), 1199-215.
Franke, Nikolaus et von Hippel, Eric (2003b), 'Satisfying heterogeneous user needs via innovation toolkits: the case of Apache security software', *Research Policy*, 32, 1199-215
Free Software Foundation 'Pourquoi «Free Software» est-il meilleur que «Open Source» ?', <http://www.gnu.org/philosophy/free-software-for-freedom.fr.html>.
Free Software Foundation 'Announcing ncurses 5.6.', <http://www.gnu.org/software/ncurses/ncurses.html>.
Free Software World Conference 'Conferencia Internacional de Software Libre 3.0', <http://www.freesoftwareworldconference.com/en/files/dossier_en.pdf>.
Fuggetta, Alfonso (2003), 'Controversy Corner: Open source software an evaluation', *The Journal of Systems and Software,* (66), 77-90.
FUNOC 'ARPANET, l'ancêtre d'Internet', <http://www.funoc.be/etic/doss001/art001.html>.
Garcia, Rosanna et Calantone, Roger (2002), 'A critical look at technological innovation typology and innovativeness terminology: a literature review', *Journal of Product Innovation Management*, 19, 110-32.
Garel, Gilles (2003a), 'Pour une histoire de la gestion de projet', *Annales des Mines : Gérer et Comprendre,* (77), 77-89.
Garel, Gilles (2003b), *Le management de projet* (Repères: La découverte).
Garel, Gilles, Giard, Vincent, et Midler, Christophe (2001), 'Management de projet et gestion des ressources humaines', (Cahier de Recherche du GREGOR), 1-32.

Garzarelli, Giampaolo (2003), 'Open Source Software and the Economics of Organization', (Università degli Studi di Roma), 1-21.

Gasse-Hellio, Matthieu (2002), 'Les tontines dans les pays en voie de développement', (Université Saint Quentin en Yvelines).

Gaudeul, Alexandre (2007), 'Do Open Source Developers Respond to Competition? The (LA)TEX Case Study', *Review of Network Economics*, 6 (2), 239-63.

Gauguier, Jean-Jacques (2005), 'Les motivations des développeurs dans l'Open Source - Une revue de la littérature', (France Telecom), 1-28.

Gawer, Annabelle (2000), 'The organization of platform leadership: an empirical investigation of Intel's management processes aimed at fostering complementary innovation by third parties (PhD thesis)', (Alfred P. Sloan School of Management, Massachusetts Institute of Technology).

Ghosh, Rishab A., Glott, Ruediger, Krieger, Bernhard, et Robles, Gregorio (2002), 'Free/Libre and Open Source Software: Survey and Study', (International Institute of Infonomics, University of Maastricht), 1-69.

Giard, Vincent (1991), *Gestion de projets* (Collection Gestion: Editions Economica).

Giard, Vincent et Midler, Christophe (1993), *Ecosip, Pilotages de Projets et Entreprises - Diversités et convergences* (Collection Gestion: Editions Economica).

Gourden, Jean-Michel (1988), *Gens de Métier & Sans-Culottes : Les Artisans dans la Révolution* (Editions Créaphis).

Gouron, André (1958), *La réglementation des Métiers en Languedoc au Moyen-Âge* (Freymond, Jacques

L'Huillier, Jacques edn.; Genève et Paris: Librairie E. Droz et Librairie Minard).

Govindarajan, Vijay et Kopalle, Praveen K. (2006), 'The Usefulness of Measuring Disruptiveness of Innovations Ex Post in Making Ex Ante Predictions', *Journal of Product Innovation Management*, 23, 12-18.

Grand, Simon, Von Krogh, Georg, Leonard, Dorothy , et Swap, Walter (2004), 'Resource Allocation Beyond Firm Boundaries: A Multi-Level Model for Open Source Innovation', *Long Range Planning*, 37, 591-610.

Grassineau, Benjamin (2009), 'La dynamique des réseaux coopératifs. L'exemple des logiciels libres et du projet d'encyclopédie libre et ouverte Wikipédia (Thèse de doctorat en sociologie)', Thèse de doctorat (Université Paris Dauphine).

Gruber, Marc et Henkel, Joachim (2006), 'New ventures based on open innovation: an empirical analysis of start-up firms in embedded Linux', *International Journal of Technology Management*, 33 (4), 356-72.

Guérin, James (2001), *Le Logiciel Libre : Précis et Concis* (O'Reilly).

Hamerly, Jim, Paquin, Tom, et Walton, Susan (1999), 'Freeing the Source: The Story of Mozilla', in Chris DiBona, Sam Ockman, et Mark Stone (eds.), *Open Sources: Voices from the Open Source Revolution* (Sebastapol, CA: O'Reilly), 197-206.

Hardin, Garrett (1968), 'The Tragedy of the Commons', *Science*, 162, 1243-48.

Hargadon, Andrew et Sutton, Robert I. (2000), 'Building an innovation factory', *Harvard Business Review*, May-june, 157-66.

Hargadon, Andrew et Sutton, Robert. I. (2003), 'Créer un laboratoire d'innovation', *Les meilleurs articles de la Harvard Business Review sur l'innovation* (Editions d'Organisation), 65-93.

Hars, Alexander et Ou, Shaosong (2002), 'Working for Free? Motivations for Participating in Open-Source Projects', *International Journal of Electronic Commerce*, 6. (3), 25-39.

Hatchuel, Armand (2001), 'Rapport de Armand Hatchuel Séance B. Kogut & A. Metiu - Séminaire Condor 18/11/1999', in Hervé Dumez (ed.), *Management de l'innovation, management de la connaissance* (Logiques de gestion; Paris: L'Harmattan), 81-83.

Hatchuel, Armand (2005), 'Séminaire d'épistémologie et méthodes quantitatives, Masters GDO-MOPP', (Paris: Mines ParisTech - Université Paris X).
Hatchuel, Armand (2007), 'Histoire des Systèmes Industriels', (Master PIC/OPEN, Ecole des Mines de Paris).
Hatchuel, Armand, Le Masson, Pascal, et Weil, Benoît (2002), 'De la gestion des connaissances aux organisations orientées conception', *Revue Internationale des Sciences Sociales de l'UNESCO,* (171), 29-42.
Hecker, Frank 'Mozilla at One: A Look Back and Ahead', <http://www.mozilla.org/mozilla-at-one.html>.
Hecker, Frank 'Setting Up Shop: The Business of Open-Source Software', <http://hecker.org/writings/setting-up-shop>.
Henderson, Rebecca (2006), 'The Innovator's Dilemma as a Problem of Organizational Competence', *Journal of Product Innovation Management,* 23, 5-11.
Henkel, Joachim (2003a), 'Software development in embedded Linux - informal collaboration of competing firms', *Internationalen Tagung Wirtschaftsinformatik* (Dresden.).
Henkel, Joachim (2003b), 'Open Source Software from Commercial Firms: Tools, Complements, and Collective Invention ', *accepted for publication at Zeitschrift für Betriebswirtschaft*.
Henkel, Joachim (2004), 'The Jukebox Mode of Innovation: a Model of Commercial Open Source Development', (Munich: University of Munich, CEPR).
Henkel, Joachim (2006), 'Selective revealing in open innovation processes: The case of embedded Linux', *Research Policy,* 35 (7), 953-69.
Henkel, Joachim (2007), 'Champions of Revealing: The Role of Open Source Developers in Commercial Firms', *European Academy of Management (EURAM)* (HEC Paris, Jouy-en-Josas).
Herbert, Robert (1941), 'The Trade Guilds Of Limerick', *North Munster Antiquarian Journal,* 2 (3).
Hertel, Guido, Niedner, Sven, et Herrmann, Stefanie (2003), 'Motivation of software developers in Open Source projects: an Internet-based survey of contributors to Linux kernel', *Research Policy,* 32, 1159–77.
Hicks, Christian et Pachamanova, Dessislava (2007), 'Back-propagation of user innovations: The open source compatibility edge', *Business Horizons,* 50, 315–24.
Hormby, Tom 'How Adobe's Photoshop Was Born', <http://web.archive.org/web/20070628002010/http://siliconuser.com/?q=node/10>.
IDC 'Worldwide Revenue from Standalone Open Source Software Will Grow 26% to Reach $5.8 Billion by 2011, IDC Research Indicates', <http://www.idc.com/getdoc.jsp?containerId=prUS20711507>.
International Organization for Standardization (ISO) 'Publication de l'ISO/CEI 29500:2008, Technologies de l'information - Description des documents et langages de traitement - Formats de fichier "Office Open XML"', <http://www.iso.org/iso/fr/pressrelease.htm?refid=Ref1181>.
Jackson, Thomas Penfield (1999), 'Findings of Fact', in United States District Court for the District of Columbia (ed.), 1-207.
Johnson, Justin Pappas (2002), 'Open source software: private provision of a public good', *Journal of Economics & Management Strategy,* 11 (4), 637-62.
Jolivet, François (1998), 'Management de projet : et si on parlait vrai ?', *Annales des Mines : Gérer et Comprendre,* (53), 19-31.
Kazakci, Akin Osman (2007), 'La théorie CKE comme fondement théorique pour les assistants de conception DesigNAR, un assistant de synthèse de concept basé sur la théorie CKE, (Thèse de doctorat en Informatique)', (Université Paris Dauphine).

Klincewicz, Krzysztof (2005), 'Innovativeness of open source software projects', (School of Innovation Management, Tokyo Institute of Technology), 11-31.
Kogut, Bruce et Metiu, Anca (2000), 'The Emergence of E-Innovation: Insights from Open Source Software Development', (Philadelphia: Wharton School, University of Pennsylvania).
Kogut, Bruce et Metiu, Anca (2001a), 'Open-Source Software Development and Distributed Innovation', *Oxford Review of Economic Policy*, 17 (2), 248-63.
Kogut, Bruce et Metiu, Anca (2001b), 'Distributed Knowledge and Open Source Software', in Hervé Dumez (ed.), *Management de l'innovation, management de la connaissance* (Logique de gestion; Paris: L'Hamattan), 23-81.
Kosse, Tim 'Filezilla History', <http://filezilla.sourceforge.net/documentation/history.htm>.
Kozinets, Robert V. (1997), '"I Want To Believe": A Netnography of The X-Philes' Subculture of Consumption', *Advances in Consumer Research*, 24, 470-75.
Kozinets, Robert V. (1998), 'On Netnography: Initial Reflections on Consumer Research Investigations of Cybtrculture', *Advances in Consumer Research*, 25, 366-71.
Kozinets, Robert V. (2002), 'The Field Behind the Screen: Using Netnography for Marketing Research in Online Communities', *Journal of Marketing Research*, XXXIX 61-72.
Kozinets, Robert V. (2006), 'Click to Connect: Netnography and Tribal Advertising', *Journal of Advertising Research*, (September), 279-88.
Kozinets, Robert V., Hemetsberger, Andrea, et Schau, Hope Jensen (2008), 'The Wisdom of Consumer Crowds. Collective Innovation in the Age of Networked Marketing', *Journal of Macromarketing*, 28 (4), 339-54.
Krishnamurthy, Sandeep (2002), 'Cave or Community? An Empirical Examination of 100 Mature Open Source Projects', (Bothell: University of Washington), 1-10.
Labatut, Julie (2009), 'Gérer des biens communs : processus de conception et régimes de coopération dans la gestion des ressources génétiques animales (Thèse de doctorat en sciences de gestion)', (Ecole des Mines de Paris).
Labbé, Frédéric (2003), 'Suite bureautique, les enjeux d'une alternative', Mémoire d'ingénieur (CNAM).
Laisné, Jean-Pierre (2006), 'Lettre ouverte / Projet de loi DADVSI et son impact potentiel sur le consortium ObjectWeb', (Consortium ObjectWeb).
Lakhani, Karim et von Hippel, Eric (2003), 'How open source software works: "free" user-to-user assistance', *Research Policy*, 32 (6), 923-43.
Langlois, Richard N. et Garzarelli, Giampaolo (2007), 'Of Hackers and Hairdressers: Modularity and the Organizational Economics of Open-source Collaboration', *European Academy of Management (EURAM)* (HEC Paris, Jouy-en-Josas.).
Lawrence, Paul et Lorsch, Jay (1989), *Adapter les structures de l'entreprise* (Collection Les Classiques; Paris: Editions d'Organisations).
Le Masson, Pascal , Weil, Benoit, et Hatchuel, Armand (2006), *Les processus d'innovation: Conception innovante et croissance des entreprises* (Stratégie et management; Paris: Hermès).
Le Monde.fr (2008), 'Chat avec Tristan Nitot (Mozilla) : "Le logiciel libre progresse à pas de géant"'. <http://www.lemonde.fr/technologies/chat/2008/05/21/quelles-evolutions-pour-le-logiciel-libre_1047610_651865.html>.
Lee, Gwendolyn K. et Cole, Robert E. (2003), 'From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case of the Linux Kernel Development', *Organization Science*, 14 (6), 633-49.
Lefebvre, Philippe (2003), *L'invention de la grande entreprise : Travail, Hiérarchie, Marché France, fin XVIIIe-début XXe siècle* (Presses Universitaires Françaises).

Leif, J. (1922), 'Préface', *Communauté et société. Catégories fondamentales de la sociologie pure.* (Les classiques des sciences sociales; Chicoutimi: Cégep, Université du Québec).

Lerner, Josh et Tirole, Jean (2000), 'The Simple Economics of Open Source', (Harvard Business School and NBER / IDEI, GREMAQ and MIT), 1-39.

Letellier, François (2008), 'Open Source Software: the Role of Nonprofits in Federating Business and Innovation Ecosystems'.

Loilier, Thomas et Tellier, Albéric (2004), 'Comment peut-on se faire confiance sans se voir ? Le cas du développement des logiciels libres', *M@n@gement*, 7 (3), 275-306.

Lorenzi, Dario et Rossi, Cristina (2008), 'Innovating through digital commons: A comparison of the innovativeness of proprietary and Free/Open Source solutions', *European Academy of Management (EURAM)* (Ljubljana).

MacCormack, Alan, Rusnak, John, et Baldwin, Carliss Y. (2006), 'Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code', *Management Science*, 52 (7), 1015-30.

MacCormack, Alan, Rusnak, John, et Baldwin, Carliss Y. (2007), 'Exploring the links between product and organizational architectures: an empirical study of open and closed source software', (Boston, Massachusetts.: Harvard Business School).

Madanmohan, T.R. et Navelkar, Siddhesh (2002), 'Roles and Knowledge Management in online technology communities: An ethnography study', (Bangalore: Indian Institute of Management Bangalore).

Maggioni, Mario A. (2002), 'Open Source Software Communities and Industrial Districts: a Useful Comparison?', (Milan, Italy: Università Cattolica del Sacro Cuore), 1-22.

Maguire, James (2007), 'The SourceForge Story'. <http://itmanagement.earthweb.com/cnews/article.php/3705731>.

Mance, Euclides André (2003), *La révolution des réseaux*, trans. Marjorie Yerushalmi (Gouvernance & démocratie; Paris: Descartes & Cie).

Markides, Constantinos (2006), 'Disruptive Innovation: In Need of Better Theory', *Journal of Product Innovation Management*, 23, 19-25.

Masson, Terrence (1999), *CG 101: a computer graphics industry reference* (New Riders Press).

Mauss, Marcel (1925), *Essai sur le don. Forme et raison de l'échange dans les sociétés archaïques.*, ed. Jean-Marie Tremblay (Les classiques des sciences sociales; Chicoutimi: Université du Québec).

Mény, Georges (1910), 'La lutte contre le sweating-system : thèse pour le doctorat', (Université de Paris, Faculté de droit, Librairies des Sciences Politiques et Sociales).

Merges, Robert P. (2004), 'From Medieval Guilds to Open Source Software: Informal Norms, Appropriability Institutions, and Innovation ', *Conference on the Legal History of Intellectual Property* (Madison, Wisconsin).

Meyer, Peter B. (2003), 'Episodes of Collective Invention', (Office of Productivity and technology, U.S. Bureau of Labor Statistics), 1-32.

MinGW Team 'Minimalist GNU for Windows', <http://www.mingw.org/>.

Mintzberg, Henry (1982), *Structure et Dynamique des Organisations*, trans. Pierre Romelaer (Editions d'Organisations).

Mockus, Audris , Fielding, Roy T., et Herbsleb, James (2000), 'A Case Study of Open Source Software Development: The Apache Server', *ACM*, 1-10.

More, Thomas (1842), *L'Utopie*, trans. Victor Stouvenel.

More, Thomas (1987), *L'Utopie ou Le Traité de la meilleure forme de gouvernement*, trans. Marie Delcourt (GF Flammarion).

Mozilla Foundation 'Mozilla Foundation Forms New Organization to Further the Creation of Free, Open Source Internet Software, Including the Award-Winning Mozilla Firefox Browser', <http://www.mozilla.org/press/mozilla-2005-08-03.html>.

Mozilla.org 'Mozilla.org announces launch of the Mozilla Foundation to lead open-source browser efforts', <http://www.mozilla.org/press/mozilla-foundation.html>.

Muselli, Laure (2004a), 'Licences informatiques et modèles d'affaires open source', *AIMS n°13* (Vallée de Seine).

Muselli, Laure (2004b), 'La licence informatique : un outil stratégique puissant pour les éditeurs de logiciels ?', (CEPN, Université Paris Nord), 1-15.

Muselli, Laure (2005), 'Open Source, création et captation de valeur', *AIMS*.

Muselli, Laure (2006), 'Entre "ouverture" et "open source" les stratégies de licence des éditeurs de logiciels', *Atelier de Recherche de l'AIMS*.

Mute Team 'How Ants Find Food', <http://mute-net.sourceforge.net/howAnts.shtml>.

NeoOffice 'Pourquoi NeoOffice est-il séparé d'OpenOffice.org?', <http://www.neooffice.org/neojava/fr/faq.php>.

Nitot, Tristan (2009), 'Open source : l'abondance du choix', *01net Pro / Avis d'expert*. <http://pro.01net.com/editorial/503645/open-source-labondance-du-choix/>.

Noll, John (2007), 'Innovation in Open Source Software Development: a Tale of Two Features', in IFIP (ed.), *Open Source Development, Adoption and Innovation* (Open Source Development, Adoption and Innovation, 234; Boston: Springer), 109-20.

Nonaka, Ikujiro (1994), 'A Dynamic Theory of Organizational Knowledge Creation', *Organization Science*, 5 (1), 14-37.

Novell Inc. (2007), 'Quarterly Report Pursuant to Section 13 or 15(d) of the Securities Exchange Act of 1934', (Washington, D.C.: United States Securities And Exchange Commission), 1-49.

Nuvolari, Alessandro (2003), 'Open Source Software Development: Some Historical Perspectives', (Eindhoven Centre for Innovation Studies), 1-29.

O'Riordan, Ciarán 'Tivoisation explained - implementation and harms, Fellowship of FSFE', <http://www.fsfe.org/en/fellows/ciaran/ciaran_s_free_software_notes/tivoisation_explained_implementation_and_harms>.

O'Mahony, Siobhán (2002), 'The emergence of a new commercial actor: community managed software projects (unpublished PhD thesis)', (Stanford University).

O'Mahony, Siobhán (2003), 'Guarding the commons: how community managed software projects protect their work', *Research Policy*, (32), 1179-98.

O'Mahony, Siobhán (2005), 'Non-Profit Foundations and their Role in Community-Firm Software Collaboration ', in Joseph Feller, Brian Fitzgerald, Scott A. Hissam, et Karim R. Lakhani (eds.), *Making Sense of the Bazaar: Perspectives on Open Source and Free Software* (The MIT Press), 393-413.

O'Mahony, Siobhán (2007), 'The governance of open source initiatives: what does it mean to be community managed?', *Journal Management Governance*, (11), 139-50.

O'Mahony, Siobhán et Ferraro, Fabrizio (2007), 'The emergence of governance in an Open source community', *Academy of Management Journal*, 50 (5), 1079-106.

Onetti, Antonio et Capobianco, Fabrizio (2005), 'Open source and business model innovation: The Funambol case', *Proceedings of the first International Conference on Open source Systems* (Scotto, M.

Succi, G. edn.; Genova).

OpenOffice.org 'IBM joins the OpenOffice.org community to develop and promote OpenOffice.org technology', <http://www.openoffice.org/press/ibm_press_release.html>.

Osterloh, Margit et Rota, Sandra (2007), 'Open source software development: Just another case of collective invention?', *Research Policy*, 36, 157-71.

OW2 Consortium 'OW2 Board Meeting, Summary of Minutes, 30 Janvier.', <http://www.ow2.org/xwiki/bin/download/About/Board/20080130-JAN08-BoardMinutesSummary.pdf >.

Pal, Nilendu et Madanmohan, T.R. (2002), 'Competing on open source: Strategies and Practises', (Bangalore: Indian Institute of Management Bangalore).

Pellarin, Charles (1843), *Charles Fourier : sa vie et sa théorie* (Librairie de l'Ecole sociétaire).

Peter, Ayton , Ferrell, William R., et Stewart, Thomas R. (1999), 'Commentaries on ‘‘The Delphi technique as a forecasting tool: issues and analysis’’ by Rowe and Wright', *International Journal of Forecasting*, (15), 377-81.

Potier, Christine et Vercken, Christine (2000), ' Qu'est-ce que la correction gamma ?', (Paris: Département d'informatique, Ecole nationale supérieure des télécommunications), 1-8.

Praca, Edwige (2003), *Histoire de la Mutualité dans l’Hérault : la grande aventure de la solidarité dans l’Hérault – XIXe et XXe siècles* (Toulouse: Editions Privat).

QualiPSo 'About QualiPSo', <http://www.qualipso.org/index.php?option=com_content&task=view&id=1&Itemid=66>.

Rannou, Hervé et Ronai, Maurice (2003a), 'Etude sur l’industrie du logiciel, Partie 3 - Annexes', (Conseil Stratégique des Technologies de l’Information).

Rannou, Hervé et Ronai, Maurice (2003b), 'Etude sur l’industrie du logiciel, Partie 2 - Rapport', (Conseil Stratégique des Technologies de l’Information).

Rastetter, Yvon (2002), *Les logiciels libres dans les entreprises* (Hermes Science Publications).

Raymond, Eric S. (1999), *The Cathedral and the Bazaar* (Sebastopol, CA: O’Reilly & Associates).

Raymond, Eric S. 'The Cathedral and the Bazaar', <http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/index.html>.

Renard, Georges (1927), *L'ouvrière à domicile* (Les cahiers de la femme: Radot).

Renaud, Hyppolyte (1845), *Solidarité. Vue synthétique sur la doctrine de Charles Fourier, deuxième édition* (Paris: Librairie sociétaire).

Robinet, Jean-Baptiste-René (1733-1777), 'Dictionnaire universel des sciences morale, économique, politique et diplomatique ; ou Bibliotheque de l'homme-d'état et du citoyen', (Libraires-associés).

Robinet, Jean-Baptiste-René (1777), 'Dictionnaire universel des sciences morale, économique, politique et diplomatique ; ou Bibliotheque de l'homme-d'état et du citoyen', (Libraires-associés).

Rossi, Cristina (2009), 'Software innovativeness. A comparison between proprietary and Free/Open Source solutions offered by Italian SMEs', *R&D Management*, 39 (2).

Rossi, Cristina, Bonaccorsi, Andrea, et Monica, Merito (2007), 'Open Source and the software industry: How firms do business out of an open innovation paradigm', *European Academy of Management (EURAM)* (HEC Paris, Jouy-en-Josas).

Rowe, Gene et Wright, George (1999), 'The Delphi technique as a forecasting tool: issues and analysis', *International Journal of Forecasting*, 15, 353–75.

Royer, Isabelle (2005), 'Le Management de Projet : Evolutions et Perspectives de Recherche', *Revue Française de Gestion*, 31 (154), 113-22.

Rullani, Francesco (2007), 'Dragging developers towards the core: How the Free/Libre/Open Source Software community enhances developers' contribution', *European Academy of Management (EURAM)* (HEC Paris, Jouy-en-Josas).

Sardas, Jean-Claude (2006), 'Equipes pluri-métiers de conception et Rationalisation de la conception, Master Gestion et Dynamique des Organisations', (Paris: Mines ParisTech - Université Paris X).

Scheid, François (2007), 'Les innovations radicales sont-elles conçues par les utilisateurs ? Le cas d'un éditeur de logiciel', *Réseaux*, 4 (143), 149-73.

Scheid, François (2009), 'Rôle des premiers clients dans la conception d'innovation radicale : le cas du logiciel (Thèse de doctorat en sciences de gestion)', (Centre de Recherche en Gestion, Ecole Polytechnique).

Schewe, Jeff 'Thomas & John Knoll', <http://photoshopnews.com/feature-stories/photoshop-profile-thomas-john-knoll-10/>.

ScummVM 'GPL conflict with Atari', <http://www.scummvm.org/news/20090616/>.

Segrestin, Blanche (2004), 'Les partenariats d'exploration : des pratiques inédites en quête d'outils et de statuts', (Centre de Gestion Scientifique, Ecole des Mines de Paris).

Segrestin, Blanche (2006), *Innovation et coopération interentreprises - Comment gérer les partenariats d'exploration ?* (CNRS Economie: CNRS Editions).

Séguin, Laurent 'Rencontre avec Eric Bachard du projet OpenOffice.org', <http://www.framablog.org/index.php/post/2009/06/12/eric-bachard-openofficeorg>.

Shah, Sonali K. (2006), 'Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development', *Management Science*, 52 (7), 1000-14.

Shaikh, Maha et Cornford, Tony (2003), 'Version Management Tools: CVS to BK in the Linux Kernel', *International Conference on Software Engineering* (Portland, Oregon), 127-31.

Sharma, Srinarayan, Sugumaran, Vijayan, et Rajagopalan, Balaji (2002), 'A Framework for Creating hybrid- open source software Communities', *Information Systems Journal*, 12 (1), 7-26.

Slater, Stanley F. et Mohr, Jakki J. (2006), 'Successful Development and Commercialization of Technological Innovation: Insights Based on Strategy Type', *Journal of Product Innovation Management*, 23, 26-33.

Smith, Adam (1776), *Recherche sur les causes et la nature de la richesse des Nations* (Tremblay, Jean-Marie edn., Les classiques des sciences sociales; Chicoutimi: Cégep, Université du Québec).

Spaeth, Sebastian, von Krogh, Georg, Stuermer, Matthias, et Haefliger, Stefan (2007), 'Lightweight Reuse In Open Source Software', *European Academy of Management (EURAM)* (HEC Paris, Jouy-en-Josas).

Spencer, Herbert (1877), *The principles of sociology* (Williams and Norgate).

Spiller, Dorit et Wichmann, Thorsten (2002), 'FLOSS Final Report - Part 3: Basics of Open Source Software Markets and Business Models', (Berlecon Research GmbH), 1-58.

Stallman, Richard 'Free Unix', <http://www.gnu.org/gnu/initial-announcement.html>.

Stallman, Richard (2002), 'Obstructing Custom Adaptation of Programs', in Joshua Gay (ed.), *Free Software, Free Society: Selected Essays of Richard M. Stallman* (Boston: Gnu Press), 124-25.

Stevenson, Robert L. (1984), *Un mort encombrant*, trans. Pierre Leyris (Collection Hachette Jeunesse: Hachette Livre).

Stuermer, Matthias, Sebastian, Spaeth, et von Krogh, Georg (2009), 'Extending private-collective innovation: a case study', *R&D Management*, 39 (2), 2009.

Tellis, Gerard J. (2006), 'Disruptive Technology or Visionary Leadership?', *Journal of Product Innovation Management*, 23, 34-38.

The Economist (2006), 'Open-source business. Open, but not as usual', (March 16th).

The Linux Information Project 'An Introduction to Tivoization', <http://www.linfo.org/tivoization.html>.

Theuriet, André (1891), *Le Mari de Jacqueline* (Collection Bibliothèque Charpentier: G. Charpentier & E. Fasquelle) 254-73.
Thomas, Cédric (2007), 'Leading Open Source Middleware OW2. OW2 General Presentation', (OW2 Consortium).
Thomas, Cédric (2009), 'Platform Strategies and the OW2 Consortium', (OW2 Consortium), 1-18.
Tönnies, Ferdinand (1922), *Communauté et société. Catégories fondamentales de la sociologie pure.*, trans. J. Leif (Les classiques des sciences sociales; Chicoutimi: Cégep, Université du Québec).
Torres-Blay, Olivier et Gueguen, Gael (2003), 'Linux contre Microsoft : La guerre des Ecosystèmes d'Affaires', (Cahier de Recherche EM Lyon), 1-32.
Transmeta Corporation 'What is Linux?', <http://web.archive.org/web/19980130085039/http://www.kernel.org/>.
Tuomi, Ilkaa (2001), 'Internet, Innovation, and Open Source: Actors in the Network', *First Monday*, 6 (1).
Tuomi, Ilkka (2005), 'The Future of Open Source', in Marleen Wynants et Jan Cornelis (eds.), *How Open is the Future? Economic, Social & Cultural Scenarios inspired by Free & Open-Source Software* (Brussels: VUB Brussels University Press), 429-59.
Ulhoi, John P. (2004), 'Open source development: a hybrid in innovation and management theory', *Management Decision*, 42 (9), 1095-114.
University of Michigan (1998-1999), 'Family Ties Inspire Alumnus to Develop the World's Most Popular Photo Software', *Engineer On line Edition*, (Fall/winter). <http://web.archive.org/web/20000416081548/http://www.engin.umich.edu/engineer/9899FW/impact/spotlight.html>.
Valette, Jacques (1980), 'Préface', *Victor Considérant (1848), Description du Phalanstère et Considération Sociales sur l'Architectonique* (Paris-Genève: Ressources).
Valette, Jacques (1981), 'Utopie sociale et utopistes sociaux en France vers 1848', in John Bartier et Jacques Valette (eds.), *Les utopismes sociaux - Utopie et action à la veille des journées de février* (Paris: CDU et Sedes réunis), 11-110.
Välimäki, Mikko (2003), 'Dual Licensing in Open Source Software Industry', *Systemes d´Information et Management*, 8 (1), 63-75.
Vanneuville-Zerouki, Edith (1999), 'Micro credits, Espaces Francophones, Espaces Anglophones, Espaces Vernaculaires', *Seminar on Employment and Sustainable Livelihoods of Persons with Disabilities ; issues in technology transfer, microcredit and institutional development* (United Nations).
Vernus, Michel (1998), *Victor Considérant 1808-1893 : Le Coeur et la Raison* (Dole: Canevas Editeur).
VideoLAN Project 'libdvdcss', <http://www.videolan.org/developers/libdvdcss.html>.
von Hippel, Eric (1988), *The Sources of Innovation* (New York: Oxford University Press).
von Hippel, Eric (2001), 'Innovation by User Communities: Learning from Open-Source Software', *MIT Sloan Review*, 42 (4), 82-86.
von Hippel, Eric (2002), 'Open source projects as horizontal innovation networks by and for users', (MIT), 1-26.
von Hippel, Eric et Katz, Ralph (2002), 'Shifting Innovation to Users via Toolkits', *Management Science*, 48 (7), 821-33.
von Hippel, Eric et von Krogh, Georg (2003), 'Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science', *Organization Science*, 14 (2), 209-23.
von Krogh, Georg (2006), 'The Promise of Private-Collective Innovation', *European Academy of Management (EURAM)* (Oslo).

von Krogh, Georg et Spaeth, Sebastian (2007), 'The open source software phenomenon: Characteristics that promote research', *Journal of Strategic Information Systems*, 16, 236-53.

von Krogh, Georg, Spaeth, Sebastian, et Lakhani, Karim (2003a), 'Community, Joining, and Specialization in Open Source Software Innovation: a case study', *Research Policy*, 32, 1217-41.

von Krogh, Georg, Haefliger, Stefan, et Spaeth, Sebastian (2003b), 'Collective Action and Communal Resources in Open Source Software Development: The Case of Freenet', (Switzerland: Institute of Management University of St. Gallen), 1-42.

Weber, Steven (2000), 'The political economy of open source', (University of Berkeley, California).

West, Joel (2003), 'How open is open enough? Melding proprietary and open source platform strategies', *Research Policy*, (32), 1259-85.

West, Joel et O'Mahony, Siobhán (2005), 'Contrasting Community Building in Sponsored and Community Founded, Open Source Projects', *Proceedings of the 38th Annual Hawaii International Conference on System Sciences* (Waikoloa, Hawaii IEEE), 1-10.

West, Joel et O'Mahony, Siobhán (2007), 'Sponsored Open Source Communities: a Vehicle for Open Innovation?', *European Academy of Management (EURAM)* (HEC Paris, Jouy-en-Josas).

Wikipedia 'Pair à pair', <http://fr.wikipedia.org/wiki/Pair_%C3%A0_pair>.

Wikipedia 'Usage share of web browsers', <http://en.wikipedia.org/wiki/Usage_share_of_web_browsers>.

Wikipedia 'NCSA Mosaic', <http://fr.wikipedia.org/wiki/NCSA_Mosaic>.

Wikipedia 'DVD Copy Control Association', <http://fr.wikipedia.org/wiki/DVD_Copy_Control_Association>.