propriété intellectuelle
PROPICE
http://www.mshparisnord.fr/ANR-PROPICE/
communs & exclusivité

Propriété et Communs

Les nouveaux enjeux de l'accès et de l'innovation partagés

Séminaire international - Paris 25-26 avril 2013

Le logiciel libre, comme commun créateur de richesses

Pierre-André Mangolte
CEPN-CNRS

Le logiciel libre, comme commun créateur de richesses

Pierre-André Mangolte
CEPN-CNRS
p.a.mangolte@wanadoo.fr
perso.orange.fr/lepouillou

version 1 - avril 2013

Ce papier porte sur le logiciel libre analysé comme commun, c'est-à-dire comme un ensemble de ressources, plus ou moins pérennes, dont l'accès et l'usage sont partagés dans un groupe très étendu, voire l'humanité toute entière (ce qui est formellement le cas des logiciels libres). Ce commun a donné naissance à des activités de production utilisant ces ressources, et les nouveaux logiciels ainsi produits entrent la plupart du temps dans le commun ; un trait qui différencie le commun des logiciels libres de bien d'autres communs, car souvent les ressources sont partagées en termes d'usage dans une population, sans qu'il y ait pour autant naissance de formes de production spécifiques au commun et l'alimentant. Il est vrai que les ressources mises en commun ici sont des objets techniques à finalité utilitaire, destinées par leur nature même à être transformées, à donner lieu à une activité innovatrice. Le logiciel est comparable aux éléments qui composent l'ordinateur (*hardware*), mais à la différence de ces pièces et composants, il est bien plus facilement modifiable. Le logiciel (*software*) n'est en effet qu'un ensemble d'instructions écrites dans un langage de programmation, et pour le transformer, il suffit de ré-écrire son code-source.

Les licences du logiciel libre ont d'ailleurs été inventées historiquement pour permettre cette modification en toute liberté et sans obstacle juridique ; le principe de base étant l'annulation des droits exclusifs (copyright ou droit d'auteur) attribués à l'auteur du code par la loi, une loi appliquée, à partir des années 1980, de manière absolue et systématique. Cette innovation institutionnelle du logiciel libre a mis en place un nouveau système de règles qui a été effectivement donné naissance à un commun, avec l'adoption croissante de ces règles à l'échelle mondiale par un grand nombre de programmeurs et la production de nombreux logiciels libres, avec l'apparition de formes de production originales, comme les grands projets de développement logiciel communautaires que sont Linux, Debian, Gnome, KDE, Mozilla, etc., et l'intégration du logiciel libre dans les activités les plus ordinaires de l'industrie informatique la plus capitaliste. Le commun alors, comme stock de ressources partagées, est devenu de plus en plus important.

Nous allons y revenir en abordant successivement les points suivants :

(1) Les droits inscrits dans les licences, la base constitutionnelle du commun,

(2) L'accès aux ressources et l'innovation logicielle,

(3) Les développements communautaires et leur spécificité,

(4) Le rapport du commun avec l'industrie et de l'industrie avec le commun.

I. La base constitutionnelle du commun, les règles applicables au code

Le logiciel libre (*free software*) tout comme le logiciel *open source* - une expression quasi-équivalente - est directement lié à la licence qui l'accompagne. Dans celle-ci, le propriétaire du copyright sur le code abandonne tout droit d'exclusivité en accordant à l'utilisateur de son logiciel des droits d'usage particulièrement étendus.

Les licences de logiciel libre ont été définies pour contrer les stratégies propriétaires, avec interdiction de toute copie et limitation des usages, c'est-à-dire l'usage habituel, recommandé par tous les traités de gestion des droits de la propriété intellectuelle, du droit exclusif sur le code source et les versions binaires ; le logiciel étant conçu comme un « actif », dont il faut protéger et exploiter la valeur, la valeur marchande évidemment. La licence du logiciel libre, à l'inverse, pose le logiciel comme un objet technique et une valeur d'usage qu'on doit pouvoir transformer, et qui doit pouvoir circuler et entrer donc dans une forme de commun.

Dans la définition du logiciel libre introduite par Richard Stallman et la *Free Software Foundation* dans les années 1980, un logiciel est considéré comme libre si la licence accorde à l'utilisateur les quatre libertés suivantes[1] :

(1) La liberté d'exécuter le programme, pour tous les usages et sans restrictions particulières.

(2) La liberté d'étudier le fonctionnement du programme et de l'adapter à ses propres besoins, ce qui signifie pouvoir corriger des erreurs, supprimer certaines parties du code, ajouter des extensions, et fusionner son propre code (ou le code d'une tierce partie) dans les modules existants.

(3) La liberté de copier le logiciel et d'en redistribuer ses copies.

(4) La liberté d'améliorer ou transformer le programme et de publier les transformations, « *pour en faire profiter toute la communauté* ».

La redistribution du logiciel, modifié ou non (point 3 et 4), inclut le code source et les formes binaires, exécutables, du programme; une redistribution qui peut être faite gratuitement ou moyennant paiement, au choix de l'utilisateur. « *Vous êtes libre de redistribuer des copies, avec ou sans modification, gratuitement ou non, à tout le monde, partout* ». Car « l'expression *free software fait référence à la liberté et non au prix* ». Le *software* est donc *free* au sens du « *free speech* » et non au sens de « *free beer* »; et il est bien précisé qu'une licence qui interdirait ou limiterait la vente ou revente des copies ne pourrait être considérée comme une licence de logiciel libre.

1 Cf. < http://www.gnu.org/licenses/license-list.fr.html >

Droits inscrits dans les licences des logiciels libres					
	Exécuter le programme	**Copier et distribuer (en l'état)**	**Modifier, pour son propre usage**	**Modifier, combiner avec un autre code, et redistribuer**	**Clause *copyleft***
Usages, domaines d'utilisation	usage final (tel que)	partage et circulation sans modification	correction de bogues, nouvelles fonctionnalités	développement logiciel	développement logiciel
Forme du logiciel	code compilé (binaire)	code source et binaires	code source	codes sources d'origines différentes	logiciel (code source +...)
Utilisateur non programmeur	oui	oui	sans objet	sans objet	sans objet
Utilisateur-programmeur	oui	oui	oui	oui	oui

Tableau 1 : Les droits (ou libertés) du logiciel libre

L'expression *open source software* est d'apparition plus récente (1998), avec la création de l'*Open Source Initiative* et l'introduction d'une procédure de labellisation des licences comportant dix critères à respecter[2]. En pratique, les deux définitions donnent presque toujours les mêmes résultats.

Pour permettre la formation du commun et assurer sa pérennité, une clause dite *copyleft* a été inventée, qui impose la redistribution du logiciel modifié (ou non) dans les mêmes conditions que la licence d'origine, autrement dit, une licence de logiciel libre avec clause *copyleft*. Attribuer à l'utilisateur des droits étendus de copie, de transformation et de redistribution du code (les quatre libertés d'un *free software*), ou mettre le logiciel dans le domaine public, et promouvoir le partage du code, ne garantit pas en effet que ce logiciel transformé va rester dans le commun. La migration d'un logiciel du domaine public ou d'un logiciel libre (sans clause copyleft) vers le logiciel propriétaire est en effet toujours possible ; on peut en effet utiliser ce code légalement pour réaliser son propre programme, distribué ensuite uniquement en versions binaires sous licence propriétaire, le code source (ancien et nouveau) restant caché. Pour éviter cette mésaventure au logiciel libre, il fallait donc une clause spécifique. Cette clause copyleft a créé une forme de domaine public sécurisé en faveur des utilisateurs de logiciel, avec un principe d'extension continue du commun, puisque tous les ajouts, toutes les modifications du code source sous *copyleft* entrent directement dans celui-ci sans pouvoir en sortir. Ce dispositif juridique élémentaire sécurise aussi à l'avance les activités de développement et de transformation du code en installant dans la durée la

2 Cf. < http://opensource.org/licenses >. En juillet 2009, il y avait 66 licences labellisées sur le site de l'*Open Source Initiative*, dont deux seulement étaient jugées par la *Free Software Foundation* trop restrictives dans les droits accordés aux utilisateurs pour être pleinement considérées comme des licences de logiciels libres.

mise en commun systématique de tout ce qui est produit dans ces activités. La clause *copyleft* fournit alors un cadre institutionnel égalitaire et pérenne, favorable au développement prolongé des logiciels libres et à la mise sur pied de formes de production en coopération.

Pour définir le commun des logiciels libres, deux distinctions sont alors importantes : (1) La distinction entre le logiciel libre (*free software* ou *open source software*) et le logiciel non libre (propriétaire, semi-propriétaire, etc.), qui établit la frontière entre le commun et le reste; (2) La distinction, interne au commun des logiciels libres, entre les logiciels sous *copyleft* et les logiciels non-copylefts, car la clause *copyleft* détermine la manière dont le code peut être redistribué.

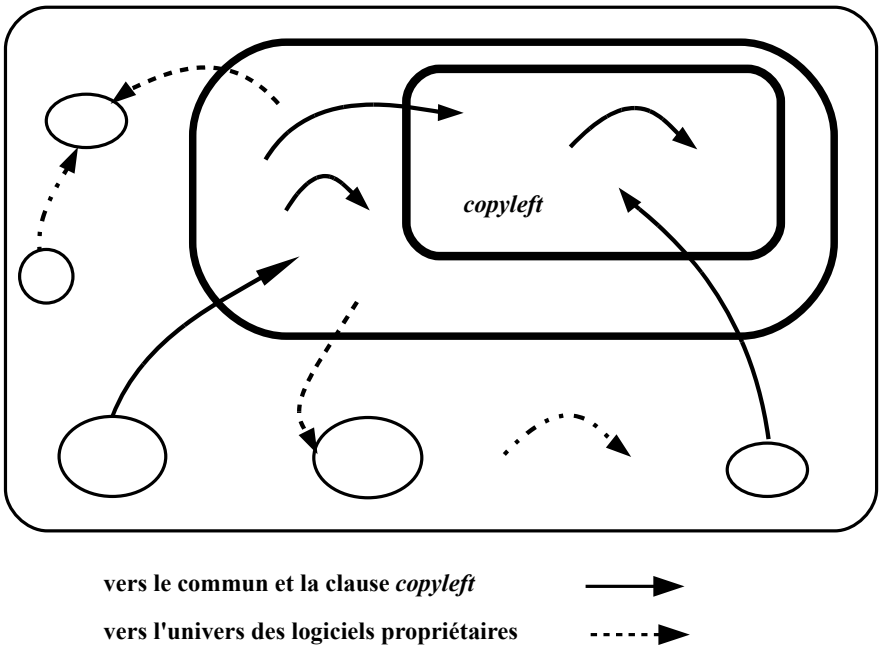

Schéma 2 : Le sens des migrations du code, d'un système de licence à l'autre

Il y a alors trois systèmes de règles en présence : (1) le système propriétaire, l'utilisation habituelle des « dpis » sur le mode de l'exclusivité, (2) le système du logiciel libre sans *copyleft*, et (3) le système du logiciel libre avec *copyleft*.

Un paquet de code tiré du commun peut servir, si la licence est sans *copyleft* (licence BSD par exemple), à construire un logiciel propriétaire, dont le code transformé échappe alors au commun ; il peut aussi servir à construire un autre logiciel libre, avec clause *copyleft* éventuellement (mise sous licence GNU-LGPL ou GPL par exemple). Mais le code copylefté ne peut être utilisé légalement que pour produire un logiciel libre avec clause copyleft ; il ne peut servir à produire un logiciel propriétaire ou même un simple logiciel libre (sans clause *copyleft*). La clause règle donc à l'avance le destin du code, au cours des opérations de transformation et de distribution des programmes.

Dans les évaluations statistiques que l'on peut faire de l'importance relative des licences de logiciel libre, une importance mesurée en terme d'adoption par les programmeurs, le plus frappant est la place centrale occupée par les licences de la *Free Software Foundation,* et particulièrement de la GNU-GPL avec sa clause *copyleft*[3].

3 Un comptage sur le site *Sourceforge* a montré que la GNU-GPL était en quelque sorte plebiscitée. Qu'il s'agisse de

Juridiquement le monde de l'open source est donc remarquablement unifié, alors même que chaque programmeur auteur d'un code peut rédiger sa propre licence ; une unification qui s'est d'ailleurs faite spontanément, sans l'appui d'un quelconque gouvernement.

II. Accès aux ressources et invention collective

Dans les licences de logiciel libre, les droits d'usage sont les mêmes pour tous les utilisateurs ; les licences n'évoquent qu'un utilisateur générique et abstrait. Mais la « ressource logicielle » est une valeur d'usage complexe qui admet des usages différents, avec ses deux formes deux formes matérielles (code source et versions binaires), et la « ressource logiciel libre » est une valeur d'usage encore plus complexe. Il faut distinguer ici au moins deux catégories d'utilisateurs, ceux qui, dépourvus de toute compétence en matière de programmation, n'utiliseront le programme que pour ce qu'il est capable de faire (les utilisateurs ordinaires ou finaux), et ceux qui à l'inverse sont suffisamment compétents pour, s'ils le souhaitent, retravailler le code source (les utilisateurs-programmeurs). Le tableau ci-dessus (tableau 1) des droits des utilisateurs montre d'ailleurs clairement que le logiciel libre a été conçu pour cette catégorie d'utilisateurs, la seule intéressée par les trois dernières colonnes du tableau, la seule qui peut s'engager réellement dans un développement logiciel, la seule capable d'utiliser pleinement les ressources du commun.

Ce sont bien les utilisateurs ordinaires et l'usage final des logiciels, et plus généralement la demande sociale adressée à l'économie des logiciels (et aux logiciels libres), qui ultimement explique et justifie le développement de tel ou tel programme. Mais pour notre propos, le « commun comme créateur de richesse », on peut laisser l'utilisateur-ordinaire de côté car seuls les utilisateurs-programmeurs sont capebles de produire du logiciel, et sont à l'origine de l'entrée dans le commun de nouvelles ressources, produites par des individus isolés ou travaillant en coopération[4]. C'est aussi dans cette population que se développe le plus le sentiment d'appartenir à une communauté spécifique, avec sa propre identité et ses propres valeurs à défendre.

L'importance des licences de logiciel libre réside dans le fait qu'elles donnent naissance à des situation d'invention collective au sens de Robert Allen (1983[5]), des situations où plusieurs inventeurs contribuent à la transformation progressive d'un objet technique, avec, selon Allen, deux pré-conditions : un échange libre d'informations sur les différentes innovations et la possibilité de réutiliser celles-ci, ce qui peut exister quand il n'y a ni brevet, ni secret de fabrication, soit par impossibilité de breveter et de garder secrète l'information, soit par une volonté de partage (ce qui est le cas pour les logiciels libres).

Un certain nombre de ressources logicielles sont disponibles dans le commun, et

---

l'ensemble des projets (plus de 160 000 recensés) ou des « 300 logiciels les plus populaires », le tableau est le même, respectivement 62 % et 63 % pour la GNU-GPL, et même 72 % et 74 % avec sa variante *lesser* au *copyleft* allégé, la GNU-LGPL.

4 Il faudrait ajouter ici ceux qui interviennent activement dans la production d'un logiciel sans travailler le code source, qui rédigent par exemple ou traduisent la documentation, ou les fichiers d'aide accompagnant le programme, etc. Le projet de développement KDE (un environnement de bureau pour les systèmes d'exploitation GNU/Linux et BSD) a environ 800 contributeurs, dont 300 pour l'équipe de traduction.

5 Robert C. Allen, « Collective invention », *Journal of Economic Behavior and Organization*, 4 (183), 1-24.

chaque programmeur peut les utiliser pour produire autre chose. Si le logiciel transformé n'est pas distribué, les licences n'imposent pas de publication. On a alors affaire à ce que la *Free Software Foundation* appelle du « *logiciel privé* », c'est-à-dire « *un logiciel développé par un utilisateur (individu, organisation ou entreprise) qui le garde pour lui et ne publie ni les fichiers sources, ni les fichiers binaires* »[6]. C'est cependant une sorte de logiciel libre « *au sens le plus trivial* », car l'unique utilisateur a tous les droits dessus. Dans cette situation, il n'y a aucun apport au commun, mais rien n'est retranché à celui-ci. Si le nouveau logiciel est distribué, transmis à d'autres utilisateurs d'une manière ou d'une autre, et si la licence d'origine comporte une clause *copyleft*, le code-source doit être fourni au destinataire de la distribution, qui peut à son tour le fournir à d'autres. Le logiciel va donc circuler d'un utilisateur à l'autre, en fonction de l'intérêt qu'il présente pour les uns et les autres. Cette circulation informelle entre utilisateurs-programmeurs est portée à un niveau supérieur par la mise en ligne, le code étant posté sur un site web, afin que tous ceux qui le souhaitent puissent le télécharger ; cette mise en ligne peut devenir permanente, ce qui est quasiment la règle pour tous les grands projets de développement. Il existe par ailleurs des dépôts (ou forges) qui rassemblent le code-source (versions stables ou en développement) de très nombreux logiciels libres, et rendent ce code accessible à toute la population des programmeurs, quelque soit l'endroit où ceux-ci se situent.

On a bien alors une situation où le code peut circuler librement avec tous les droits d'usage que confère la licence. Les conditions de l'invention collective sont donc réunies, donnant naissance à une dynamique particulière de création, d'accumulation et de partage des savoirs dans le domaine des logiciels, avec trois grandes formes d'innovation, trois modes d'évolution des ressources du commun :

(1) **Le mode cumulatif, par transformation progressive du même logiciel**

Le logiciel est modifié par des ré-écritures successives, par ajout ou suppression de portions du code, dans le but de corriger une erreur (bogue), d'améliorer la performance, d'introduire de nouvelles fonctionnalités ou d'adapter le logiciel à une transformation de l'environnement. Avec un minimum d'organisation de la circulation du code transformé, le développement logiciel se structure, pris en charge par une certaine « communauté » d'utilisateurs-programmeurs, lesquels travaillent en parallèle ou successivement à son évolution cumulative, et le logiciel final est alors le résultat de cette production collective.

(2) **La ré-utilisation et la combinaison de codes d'origines différentes**

On peut reprendre tout ou partie du code-source d'un logiciel libre pour fabriquer tout-à-fait autre chose. On peut même combiner des codes-sources d'origines différentes, à la seule condition que les licences soient compatibles entre elles. Cette liberté permet la ré-utilisation de composants logiciels existants, déjà développés et souvent bien débogués. La production par combinaison, avec l'ajout d'un nouveau code écrit pour l'occasion, est techniquement facilitée par le caractère modulaire de la plupart des logiciels libres, qui fait que l'ensemble du code-source est découpé en sous-ensembles ayant chacun leur fonction. Ces modules peuvent être ré-employés ailleurs,

6 C'est une situation fréquente en matière de production de code. Un très grand nombre de programmeurs travaillent en effet dans des organisations ou des entreprises qui ne distribuent et ne vendent aucun logiciel, comme par exemple les banques, les compagnies d'assurances, les compagnies aériennes, le métro, les universités, etc. On a alors de l'auto-production dans le périmètre de l'organisation, pour les besoins de sa propre activité (et le code ne sera que très rarement re-distribué).

pour la tâche spécifique qu'ils sont capables de remplir, et ce ré-emploi est une source d'économies non négligeables pour l'innovation en logiciel libre, car cela évite d'avoir à ré-écrire à chaque fois tous les composants nécessaires.

On peut donner un exemple de cette possibilité de ré-emploi en listant tous les programmes produits à partir des « moteurs de rendu html », un composant important des navigateurs web qui assure l'affichage correct du contenu des pages web. Il faut se rappeler qu'à la fin des années 1990, la firme *Microsoft* avait obtenu un quasi-monopole en matière d'accès à l'internet avec son navigateur Internet Explorer (95 % des utilisateurs internautes). Aujourd'hui, la situation est complètement différente et le logiciel libre n'y est pas étranger. En effet, il s'est avéré être un puissant vecteur de création de variétés et d'alternatives technologiques, et par là de dé-monopolisation, plus efficace que toutes les procédures antitrust[7]. Derrière cette variété, il y a cependant des éléments communs, avec ici trois moteurs de rendus html en logiciel libre, *Gecko, KHTML et Webkit.*

Le moteur *Gecko* est un produit de la fondation *Mozilla,* publié sous trois licences (MPL, GNU-GPL et GNU-LGPL), ce qui permet à une grande palette d'utilisateurs d'intégrer Gecko dans des produits libres ou propriétaires (dans ce cas la fermeture du code n'affecte pas le moteur qui reste toujours dans le commun). On trouve Gecko dans plusieurs navigateurs (Firefox, Camino, K-Meleon, Epiphany et Kazehakase), et dans d'autres produits (logiciels de messagerie, éditeurs de code html, etc.). Le moteur *KHTML*, sous GNU-LGPL, est un développement logiciel du projet d'environnement de bureau KDE pour les distributions GNU/Linux. Ses qualités lui valurent d'être retenu par *Apple* pour son futur navigateur Safari. Les programmeurs de la firme retravaillèrent le code afin de l'améliorer (support javascript) et de l'adapter au langage utilisé par l'interface des macs[8] . A l'époque, les développements de KHTML par la communauté KDE et de Webkit par les programmeurs d'Apple, étaient complètement séparés (ce qui représente un *fork*, voir ci-après). Le logiciel était développé chez Apple de manière privée, le code ne circulant que dans les limites de la firme sans publication ni communication à l'extérieur. Mais après la sortie de Safari, le code transformé a bien été publié et communiqué à la communauté ; et certaines modifications introduites par les développeurs d'Apple ont été intégrées dans KHTM améliorant son fonctionnement. Depuis, des variantes de Webkit sont apparues, comme WebkitGTK+ et QtWebkit, ce qui rend plus facile l'intégration de ce moteur dans d'autres environnements de bureau que Mac OS X ; et un grand nombre de navigateurs (ainsi que d'autres programmes) ont été créés intégrant Webkit. On trouve ici Safari, OmniWeb, Midori, Epiphany, Konqueror (en option), Arora, GoogleChrome (et sa version *open source* Chromium), et bien d'autres applications.

(3) **Les fourches, les écritures parallèles, et l'exploration tous azimuts**

Dans l'univers des logiciels libres, n'importe quel utilisateur peut lancer son propre projet logiciel en reprenant le code-source d'un logiciel en développement. Les développeurs de ce programme ne peuvent s'y opposer. La liberté d'entreprendre est

7 Selon *StatCounter* (août 2012), les parts mondiales (en termes d'utilisation) étaient respectivement : Internet Explorer : 32,8 %, Chrome : 33,6 %, Firefox : 22,8 %, Safari : 7,3 %, Opera : 1,6 %, autres : 1,7 %.

8 Une des limites de KHTML était le faible support du javascript, un langage de plus en plus présent dans les pages web. KHTML avait aussi été conçu pour l'environnement KDE (langage et bibliothèques Qt). Il fallait donc modifier certaines parties du code pour adapter celui-ci au langage de l'interface de Mac OS X, l'objective-C.

donc totale. Les licences mettent en effet tout le monde sur un pied d'égalité, en concurrence, et ceci en permanence. Cette situation offre la possibilité à des programmeurs motivés et compétents de relancer un développement logiciel abandonné, ou insuffisamment dynamique, ou mal géré par ceux qui en ont la charge. C'est un phénomène nommé « embranchement » ou « fourche » (en anglais *fork*). Le développement du logiciel se scinde alors, avec deux projets parallèles et concurrents, et éventuellement par la suite des emprunts de code d'un projet à l'autre. Ces développements peuvent ensuite diverger progressivement et se séparer définitivement, ou à l'inverse fusionner. Plus fréquemment, l'un des développements finit par supplanter l'autre, qui disparaît faute de contributeurs.

Les raisons des scissions et divisions peuvent être des plus diverses : conflits individuels, divergences de point de vue en matière de choix techniques, insatisfaction devant la conduite d'un projet en cours, ou un enjeu plus stratégique, comme un changement de licence ou d'orientation du développement. On peut donner un exemple récent de ce cas de figure, avec les différents *forks* apparus à la suite du rachat de *SUN Microsystems* par *Oracle*.

a) *Oracle* ayant décidé d'abandonner le développement d'*OpenSolaris,* le projet *Illumos* est apparu avec comme objectif la construction d'un système d'exploitation entièrement libre, compatible à 100% avec tous les logiciels produits antérieurement pour OpenSolaris.

b) En rachetant *SUN*, *Oracle* a aussi racheté MySQL, un système de gestion de base de données (SGBD), distribué sous une double licence GNU-GPL et licence propriétaire. Ce système est le plus utilisé au monde[9]. En mai 2009, le fondateur de MySQL (Michael Widenius) quitte alors la société pour lancer sa propre version concurrente de MySQL. Cela donne un nouvel embranchement (sous licence GNU-GPL).

c) En septembre 2010, des contributeurs de la suite bureautique OpenOffice.org créent de leur côté *The Document Foundation* afin de poursuivre le développement de ce programme, rebaptisée LibreOffice (et mis sous licence GNU-LGPL). Ils obtiennent le soutien de la *Free Software Foundation*, de *Newell*, *RedHat*, *Canonical* (distribution Ubuntu), *Google*, *Gnome*, *NeoOffice,* et d'un grand nombre de communautés formées autour d'OpenOffice.org. Un autre projet rival, abrité par la fondation *Apache* et soutenu par *IBM,* est apparu peu après sous le nom d'Apache OpenOffice. Il existe donc à l'heure actuelle deux projets de développement parallèles et concurrents qui partent du même code-source et sont tous deux des logiciels libres.

9 Ce rachat mettait dans le même panier les deux produits concurrents que sont Oracle Database et MySQL. Il a néanmoins été autorisé par la Commission européenne le 21 janvier 2010, et par les autorités antitrust américaines.

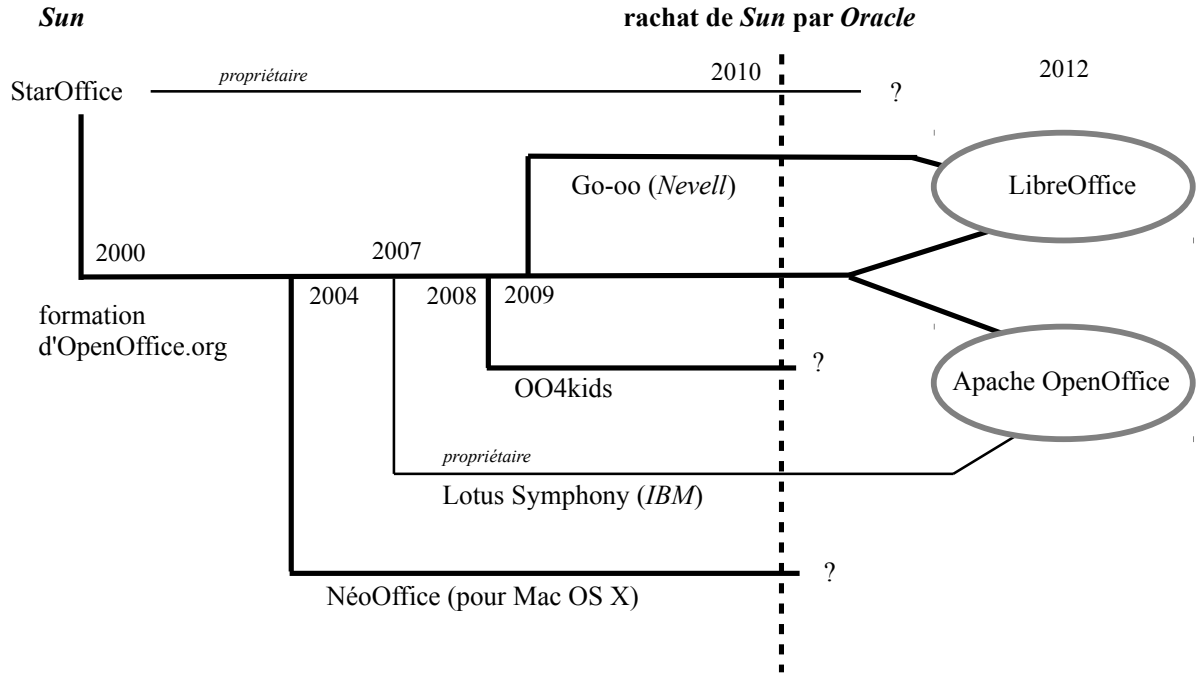

Schéma 4 : Les fourches (*forks*) d'OpenOffice.org

Les fourches sont pour le commun un moyen puissant de maintenir les règles et les droits existants (en dehors de toute clause *copyleft* et de toute action judiciaire). Quand un développement logiciel pris en charge par une équipe ou une communauté d'utilisateurs-programmeurs subit une dérive qui, de proche en proche, pourrait transformer le logiciel libre en un logiciel moins libre, voire non libre (par fermeture du code ou difficulté d'accès à celui-ci, changement de licence, etc.), la formation d'une fourche et d'un nouveau développement logiciel plus clairement ancré du côté du logiciel libre est toujours possible. C'est le sens de la formation de LibreOffice.

Mais plus profondément, c'est aussi une très grande liberté en matière d'innovation et d'exploration de différentes alternatives technologiques, quand ces alternatives existent. L'existence des fourches, des développements parallèles et concurrents, permet aux innovateurs d'explorer en parallèle les voies les plus divergentes. Quand, dans le développement d'un logiciel libre, les développeurs rencontrent un problème technique qui pose un problème de choix pour l'orientation du travail de programmation, en cas d'opinions divergentes, le jeu reste ouvert, et l'exploration d'une solution dédaignée par une partie des développeurs est toujours possible pour les autres. Les deux voies seront alors explorées et, les résultats obtenus faisant le tri, les solutions non valables finiront par être rejetées. L'existence de cette liberté de former son propre embranchement permet ainsi d'éviter aux logiciels libres de se retrouver dans des impasses technologiques. Dans certains domaines, l'innovation est ainsi tout-à-la fois redondante et considérablement diversifiée, avec des composants logiciels nombreux et disponibles, alors même que leur utilisation effective est souvent très faible. Mais si la situation vient à changer et pose un problème d'adaptation, le problème a déjà été exploré et les éléments de solution existent. L'arrivée des *netbooks* n'a ainsi posé aucun problème au

logiciel libre[10]; des solutions légères, adaptées à de petites configurations en terme de mémoire étaient en effet déjà présentes, toutes prêtes, dans le commun.

III. Les développements logiciels communautaires

Quand on observe l'évolution de la production des logiciels libres et les entrées dans le commun en longue période, on voit se dessiner une sorte de séquence. Sont apparus tout d'abord les outils logiciels indispensables aux programmeurs pour leur travail de programmation : compilateur (GCC), éditeurs de code, etc. Puis sous l'impulsion de la *Free Software Foundation*, les éléments constitutifs d'un système d'exploitation entièrement composé de logiciels libres se mettent progressivement en place. Le noyau du système, indispensable au projet d'ensemble, surgit tout-à-fait indépendamment de la FSF, avec l'essor du développement communautaire de Linux. Au milieu des années 1990, les qualités propres de Linux sont reconnues non seulement par les partisans du logiciel libre mais aussi par une partie de l'industrie. Celle-ci mesure alors l'importance et les potentialités du développement communautaire distribué. A l'époque cependant, le système GNU/Linux et tous les logiciels libres qui l'accompagnent ne sont utilisés qu'en ligne de commande, au moyen du *shell* ; les différents composants nécessaires à la gestion d'une interface graphique utilisateur (*Graphical User Interface*) n'existe pas encore en logiciel libre, et les utilisateurs du commun ne sont pour l'essentiel que des utilisateurs-programmeurs.

La période suivante, un peu avant l'an 2000, voit les premières libération de code, avec *Netscape* et le projet Mozilla, et l'adoption de Linux par une partie de l'industrie, en remplacement des Unix propriétaires qui sont abandonnés l'un après l'autre. Linux trouve aussi sa place dans le code embarqué, les serveurs, la conduite de certains processus industriels, et l'ingénierie de certains systèmes informatiques, comme les supercalculateurs. Les sociétés de services en ingénierie informatique (SSII) découvrent les avantages du libre, et les premières sociétés commerciales vendeuses de logiciels libres et de services associés apparaissent. Parallèlement, les communautés du libre s'attaquent au problème de l'interface graphique, afin de doter le système d'exploitation GNU/Linux d'un environnement de bureau ; ce sont les projets KDE (1996), Xfce (1996), Gnome (1997), etc., et d'atteindre ainsi le grand public peu habitué à piloter un ordinateur en ligne de commande. Les premières distributions GNU/Linux apparaissent sous forme de projets communautaires : Slackware (1992), Debian (lancé en 1993), Gentoo, suivies par des distributions commerciales pilotées par des firmes : RedHat (1994), Mandriva Linux (1998), Suse (1994, appartenant aujourd'hui à *Novell*), etc. C'est une nouvelle sorte de production, visant à constituer une collection de logiciels, assemblée en un tout cohérent autour d'un noyau Linux. Ces distributions sont destinées tant aux utilisateurs-programmeurs qu'aux utilisateurs ordinaires, avec des choix techniques et des philosophies différentes. Elles s'adressent ainsi à toutes les catégories d'usages, et sont de fait très nombreuses (plus de 500 aujourd'hui).

Depuis, cette production s'est encore diversifiée, et les logiciels libres envahissent

10 Ce qui ne fut pas le cas pour *Microsoft*, dont le système (Vista) trop gourmand en ressources demandait des machines plus rapides. Microsoft a du alors en catastrophe, pour ne pas perdre ce nouveau marché où les ventes étaient en pleine expansion, ressortir XP dans une version allégée, un produit pourtant abandonné et officiellement déclaré obsolète.

progressivement les différents domaines de l'économie et de la société où le logiciel est présent, en s'installant sur toutes les plates-formes, y compris les plus récentes, comme les smarphones et tablettes, et dans toutes les niches en matière d'usages.

Au cœur de cette évolution et cette utilisation du commun à des fins productives, il y a les développements communautaires qui mobilisent un grand nombre de programmeurs travaillant en coopération. A l'origine, on a toujours une certaine masse de code et un certain nombre de problèmes techniques, qui font qu'un individu ou une petite équipe ne peut assurer seul le développement du programme. Un réseau ou une communauté d'utilisateurs-programmeurs prend alors en charge ce développement en coopération, une coopération rendue possible par les licences, et plus particulièrement la clause *copyleft.* Dans ce type de coopération, les relations sont formellement symétriques et égalitaires, car tous ont les mêmes droits sur le code source qu'ils transforment, lequel peut évoluer (et en principe s'améliorer) en intégrant les contributions des uns et des autres. Cette forme de production est un exemple particulier de ce que Von Hippel (2002) a appelé un « *réseau horizontal d'innovation par et pour les utilisateurs* »[11] en désignant ainsi une situation où les utilisateurs d'une technique sont à la source de la transformation de la technique, l'évolution de celle-ci étant directement pilotée par l'usage.

Les utilisateurs d'un logiciel sont en effet toujours intéressés à voir celui-ci s'améliorer, et quand ils sont programmeurs et disposent du code avec le droit de le transformer, ils peuvent le faire, afin d'obtenir une meilleure valeur d'usage. Ils produisent alors un nouveau code. Faire circuler le nouveau code n'est guère problématique à l'époque de l'internet. Le cacher par contre ne leur rapporterait rien, alors que sa publication peut permettre à d'autres d'innover à leur tour, avec au final un logiciel amélioré dont tous pourront bénéficier. On comprend alors pourquoi certains projets de développement en logiciel libre ont émergé aussi vite, dans une dynamique de coopération et d'invention collective, sans qu'il y ait ici ni salariat ni échanges marchands, les contributeurs n'étant jamais payés pour la fourniture de leur code[12].

En pratique, les projets communautaires posent des problèmes de coordination, et les projets qui se sont effectivement développés (comme Linux) sont caractérisés par l'émergence d'une administration pratique du développement, avec un grand nombre d'outils utilisant l'internet (système de gestion des versions (CVS, Git, etc.), forums, mails, système de suivi de bogues (Bugzilla, etc.)), l'apparition d'une division du travail horizontale et verticale, et une certaine différenciation et répartition des rôles, l'ensemble permettant l'organisation routinière du travail d'ensemble. La coordination des activités est réalisée en particulier par (1) un usage systématique de la modularité et (2) une administration du projet exercée par un (ou des) mainteneur(s).

(1) Le code-source du projet est découpé en différents modules de taille réduite en terme de lignes de code, lesquels sont reliés par des interfaces. Chaque module peut alors être développé séparément sans qu'il soit nécessaire que ses développeurs se coordonnent avec ceux qui interviennent sur des modules différents. Il leur suffit de respecter rigoureusement les interfaces. Cette modularité rend possible l'établissement d'une division horizontale du travail avec une spécialisation poussée de chacun, la

11 Cf. Von Hippel, Eric, 2002, « Open source projects as horizontal innovation networks – by and for users », *MIT Sloan School of Management Working Paper,* n° 4366-02 juin.

12 Même si aujourd'hui, beaucoup de développeurs de ces projets communautaires sont des salariés employés par des fondations ou des entreprises pour participer au développement de tel ou tel projet.

cohérence d'ensemble étant assurée par le respect des interfaces communes et l'architecture générale du système, plus l'utilisation des éléments communs, comme les bibliothèques. Les modules sont alors développés par des équipes réduites (parfois un programmeur isolé), et très nombreux sont les auteurs qui ne travaillent que sur quelques modules; plus de 70 % des auteurs de Linux n'ont ainsi travaillé que sur un seul module (Gosh et David, 2003[13]). C'est aussi ce qui rend possible le travail géographiquement distribué et la mobilisation en même temps d'un très grand nombre de programmeurs, sans poser de problème particulier à la coordination.

(2) Le deuxième trait caractéristique des projets *open source* est l'existence d'une division verticale du travail, avec une stratification marquée de la communauté des contributeurs. L'intensité des contributions est en effet très variable et les rôles différenciés. Le schéma général est celui d'un noyau de programmeurs relativement stable, qui écrit la majorité du code et contrôle le développement au niveau des modules, en intégrant des contributions ponctuelles issues d'un groupe plus large de développeurs, entouré à son tour par ceux qui signalent ou fixent les défauts (bogues), avec enfin la communauté bien plus large des utilisateurs ordinaires des versions (stabilisées) du logiciel. La transformation du logiciel repose toujours sur l'accès au code et un principe de contributions libres, mais l'intégration de ces contributions à l'ensemble du code n'est pas automatique. Il y a toujours un filtrage assuré par les mainteneurs responsables de tel ou tel module, afin d'obtenir au final une version améliorée du logiciel, et souvent une seule version.

Le commun pris dans son ensemble représente une sorte d'anarchie au sens premier du terme, c'est-à-dire un monde sans commandement central, sans roi, sans chef, sans autorité unique. Il existe bien ça et là des organisations comme la *Free Software Foundation, l'Open Source Initiative*, la *Fondation Apache* ou d'autres, qui par leurs actions influent sur l'évolution générale du commun, en déployant une activité spécifique de définition et de défense des règles d'ouverture et de partage du code (dans des formulations qui peuvent d'ailleurs différer l'une de l'autre). Elles contribuent à orienter les efforts de la communauté dans telle ou telle direction. Mais elles sont plurielles, et souvent en rivalité les unes avec les autres ; et aucune d'entre elles ne représente une autorité unique, aucune ne peut y prétendre, n'ayant d'ailleurs guère de moyens pour imposer cette autorité. C'est plutôt la communauté, plus exactement les membres de celle-ci, qui seuls ou en coalition gèrent, régulent et font évoluer le commun, en adoptant tel ou tel ensemble de règles et de comportements, en se mobilisant sur tel ou tel projet, telle ou telle tâche, telle ou telle cause, etc.

Il n'en est pas tout-à-fait de même au niveau des différents projets de développement logiciel, avec ici des formes d'organisation et d'administration extrêmement variées. Il y a cas très fréquent le projet mené et géré par un fondateur (ou une toute petite équipe), qui produit initialement une grande partie du code et oriente ensuite le développement du logiciel comme il l'entend, en intégrant ou non des contributions externes. C'est le modèle du dictateur supposé « bienveillant »; mais il y a aussi le modèle méritocratique et le modèle démocratique qui servent de référence à la plupart des grands projets communautaires.

Le modèle méritocratique est pour donner un exemple celui de la *Fondation*

13 Gosh, Rishab Aiyer et Paul David, 2003, « The nature and composition of the Linux kernel developer community, a dynamic analysis », < http://dxm.org/papers/licks1/licksresults.pdf >.

Apache. Le principe en est le « *gouvernement par le mérite* », un mérite apprécié en fonction de la participation au développement en terme de production de lignes de code. Le principe de sélection des individus est alors le même que celui qu'on applique au code : une grande ouverture à l'entrée suivi d'un filtrage, les différents rôles étant distingués et hiérarchisés : le simple utilisateur, le développeur (ou contributeur) externe, puis le *committer*, le seul qui a le droit d'écrire dans le code-source, etc. Un autre modèle est celui de l'association Debian, qui gère la mise sur pied de distributions libres autour des noyaux Linux ou FreeBSD, avec plus de 1000 développeurs concernés. Il est défini comme « communautaire et démocratique ». Un « *contrat social* » et une constitution règle les problèmes de fonctionnement, définissant les méthodes de prises de décision et le rôle des différents acteurs. Le projet est dirigé par un chef de projet Debian élu (et éventuellement réélu) chaque année par les membres de l'association, flanqué d'un comité technique et d'un secrétaire. Les pouvoirs du chef de projet sont cependant limités et toutes les décisions d'une certaine importance donnent lieu à un vote de la communauté. Un autre poste important est le *Release manager*, aidé par des assistants, qui a pour fonction de définir (avec la communauté des développeurs) les objectifs de la prochaine version, d'en superviser le processus, et d'en fixer les dates de sorties.

IV. Le commun et l'industrie, les entreprises et le commun

Les règles inscrites dans les licences des logiciels libres ne refusent ni la société marchande, ni le capitalisme. Elles interdisent simplement l'appropriation exclusive du code-source des logiciels libres et tout contrôle exclusif de l'usage et du développement logiciel. Les activités commerciales ne sont aucunement interdites et les entreprises, comme tout autre utilisateur, peuvent utiliser le code comme elles l'entendent, en respectant les clauses de la licence. L'économie du commun est cependant très largement non-marchande, ce qui est particulièrement vrai pour les formes communautaires de production, mais pas uniquement. Il existe en effet d'autres façons de produire du logiciel libre :

(1) La production par un individu ou une toute petite équipe travaillant en relations étroites (ce qu'on peut appeler le « modèle de la cave », en référence à l'étude de Krishnamurty de 2002[14]). Cette production artisanale et solitaire à l'opposé du modèle communautaire est bien représentée en logiciel libre. Un très grand nombre de programmes de petite taille sont produits ainsi, aujourd'hui comme hier. Il s'agit en général d'un développement engagé par un utilisateur pour ses propres besoins, ou par goût de l'innovation, ou par souci d'apprentissage personnel, ou par jeu, un travail non commandé réalisé comme un logiciel privé. Ce programme terminé (et suffisamment stable) est directement publié et mis dans le commun.

(2) La production prise en charge par une fondation, une organisations à but non

14 Krishnamurthy, Sandeep, 2005, « Cave or community ? An empirical examination of 100 mature open source projects », *First Monday* (10).

lucratif, qui réunit les fonds nécessaires à la mise sur pied d'un ou plusieurs projets de développement logiciel, des fonds qui servent à payer des programmeurs pour qu'ils se consacrent à temps plein au développement d'un logiciel, en collectant par ailleurs des contributions externes comme dans le mode communautaire. Le but de la fondation est aussi la gestion du projet lui-même et l'organisation de la diffusion du programme. Le logiciel est alors le résultat d'une coopération associant un groupe de développeurs salariés, payés par la fondation, et des contributeurs externes qui signalent les défauts (bogues) et travaillent à leur correction. Les fondations les plus connues sont la *Free Software Foundation* qui produit du code libre depuis les années 1980, la fondation *Apache*, assurant le développement du serveur web du même nom, et *Mozilla.org* (Firefox, Thunderbird, etc.) ; mais aujourd'hui la plupart des projets de développement communautaires se sont dotés de leur propre fondation, pour réunir des fonds et couvrir leurs besoins de manière autonome.

Dans ces deux formes de production comme dans le modèle communautaire, le but est la valeur d'usage et l'innovation. Il n'y a pas ou rarement vente du logiciel, et les modes de financement relèvent alors du <u>don</u> et de la <u>dépense</u> à fonds perdus, et non de l'<u>avance</u> en capital, pour reprendre une distinction classique de l'économie politique (Cf. Karl Marx, citant James Steuart[15]).

Type de ressource		exemples
Don direct d'un logiciel	en nature	« logiciel privé » publié comme logiciel libre
Don de code	en nature	libération d'un code-source anciennement sous licence propriétaire, qui devient le point de départ d'un nouveau développement en logiciel libre
Ré-utilisation du code appartenant déjà au commun	en nature	de droit dans toute licence d logiciel libre (le code étant disponible dans différentes forges)
Don d'heures de travail consacrées au développement	en nature	travail des mainteneurs et contributeurs externes dans les projets communautaires
Salaires versés par des entreprises ou des fondations à des employés développant un logiciel libre	en nature et monétaire	mainteneurs et contributeurs salariés dans certains projets *open source*
Apports monétaires de particuliers ou d'entreprises	monétaire	collectés en particulier par les fondations

Tableau 5 : Modes de financement d'une activité dédiée à la valeur d'usage

Ce ne sont pas les règles des licences qui ont conduit à faire de l'économie des logiciels libres une activité en général « non-marchande », financée par des dons, des

15 Karl Marx, Le capital, Livre I, 1, p. 151 et suivantes. Comparant la formule de la circulation M – A – M et la formule générale du capital A – M – A, Marx, citant James Steuart (1805), distingue entre les *avances* (les investissements) et les *dépenses*, dernier terme de la formule de la circulation, « *où l'argent est enfin converti en marchandise qui sert de valeur d'usage; il est donc définitivement - <u>dépensé</u>. Dans la forme inverse A – M – A , l'acheteur donne son argent pour le reprendre comme vendeur.* (…) *S'il le laisse partir, c'est seulement avec l'arrière-pensée perfide de le rattraper. Cet argent est donc simplement - <u>avancé</u>* » (p. 153).

subventions, des dépenses à fonds perdus. C'est plutôt la nature numérique des ressources logicielles et la mise en concurrence de tous les utilisateurs qui a conduit à cette situation. C'est seulement d'ailleurs dans le cas d'une distribution utilisant l'internet, que la possibilité de diffuser quasi-gratuitement, avec un coût de duplication pratiquement nul, s'est transformée en une sorte d'obligation, puisque copier et rediffuser sont un droit tout utilisateur. Pour une activité purement capitaliste, c'est une sorte de contrainte, mais aujourd'hui, ce monde non-marchand des logiciels libres et le monde capitaliste de l'industrie ont trouvé leur articulation. Les modèles économiques commerciaux des entreprises ont été reconfigurés de manière à articuler la formule générale du capital et la recherche de la rentabilité, et la dépense avec le financement à fonds perdus d'un certain nombre de projets de développement logiciel, souvent communautaires, en rapport plus ou moins étroit avec la production des entreprises concernées ; et il n'y a pas de problème majeur ici.

Les entreprises utilisatrices du commun sont d'ailleurs comme les autres utilisateurs. Certaines sont dans la situation d'un utilisateur ordinaire. Elles ont besoin de logiciels pour assurer le fonctionnement routinier de leurs activités, sans être engagées dans un développement logiciel particulier. D'autres vont utiliser le logiciel libre pour construire leur offre commerciale. Ce qui les intéresse alors est la valeur commerciale que ces logiciels peuvent acquérir une fois intégrés dans une offre plus large (avec en plus des économies en matière de développement). Elles vendent le logiciel libre tiré du commun, obtenu souvent « gratuitement », en y ajoutant d'autres produits ou services. On peut citer ici les distributions commerciales GNU/Linux, une offre qui intéresse surtout d'autres entreprises, grandes ou petites[16]. D'autres firmes ont simplement renouvelé une offre vieillissante, en abandonnant les Unix propriétaires pour Linux et tous les logiciels libres qui l'accompagnent. C'est le choix d'*IBM* et de bien d'autres, le recours aux logiciels libres ayant un autre avantage, « stratégique », l'abolition de fait de tout droit de propriété intellectuelle exclusif, et de tout ce qu'il porterait en lui comme possibilité de verrouillage technico-juridique, de création de relation de monopole, de situation d'enfermement, et de mises à rançon éventuelle ; ce qui facilite l'établissement des relations commerciales dans une grande partie de l'industrie. Des alliances se sont alors nouées entre ces firmes capitalistes et les développements communautaires ; les entreprises contribuent au financement de ces développements ou y engagent une partie de leurs salariés ; des firmes concurrentes coopèrent alors ensemble sur des projets communs, ce qui n'a d'ailleurs rien d'exceptionnel. En termes de management, on parlera simplement « *d'une mutualisation des ressources* », car certains logiciels libres sont devenus de fait des ressources communes indispensables à toute une partie de l'industrie.

D'autres entreprises par contre sont intéressées par la liberté que donne au programmeur le logiciel libre en matière de ré-utilisation et de transformation du code-source. C'est le cas assez général des *SSII*, spécialisées en développement logiciel, en ingénierie de système, en production de logiciels sur commande, etc. Les solutions en logiciel libres qu'elles fournissent et facturent peuvent intégrer alors des ressources

16 Ces offres commerciales sont souvent flanquées d'une distribution en gratuiciel réservé aux utilisateurs domestiques, et d'un projet de développement sous forme communautaire, avec ici un retour d'expériences et l'apport de contributeurs externes (tests et correction de bogues, etc.) que les relations purement commerciales ne permettraient pas d'assurer. Ainsi *Red Hat* et le projet Fedora, *Suse* et OpenSuse, etc.

tirées du commun. Le code est fourni sous une licence de logiciel libre au client, les recettes obtenues payant le travail de développement, les coûts de fonctionnement de l'entreprise et la rémunération des capitaux. On est bien alors dans la formule générale du capital. Au niveau le plus élémentaire, on trouve ici la production de code embarqué (avec Linux), un code dédié, intégré dans une machine particulière (ascenseur, téléphone portable, routeur, camescope, etc.), mais la même recherche de ressources utilisables pour l'innovation explique aussi l'importance des logiciels libres dans la construction des machines les plus puissantes, les supercalculateurs[17], etc.

17 Aujourd'hui, la majorité de ces machines utilisent Linux (92,4 % des 500 supercalculateurs les plus puissants en juin 2012) et d'autres logiciels libres (*Lustre* par exemple, sous licence GNU-GPL). C'est d'ailleurs souvent une exigence des clients, le développement logiciel devant être poursuivi après livraison par l'utilisateur, dans la recherche d'une amélioration continue des performances de la machine.