LOGICIELS LIBRES ET LE TRAVAIL INFORMATIQUE DANS LE CAPITALISME DE LA CONNAISSANCE

Pablo Miguez
Université de Buenos Aires, Argentina

Résumé.- Ce travail s'est proposé l'analyse des conséquences du travail immatériel dans la production de logiciels dans le cadre de la citoyenneté numérique. L'étude s'est portée sur les relations entre les travailleurs informatiques dans la production de logiciels et les services informatiques à partir de la comparaison entre deux modèles de développement de logiciels, l'un inspiré des logiciels libres et l'autre qui dérive des logiciels fermés. Le travail des travailleurs informatiques fut analysé à travers le corpus théorique préalable au mouvement des logiciels libres. Certains de ces principaux représentants font un usage singulier des catégories de l'anthropologie économique, dont la pertinence est remise en question au regard de l'analyse de l'organisation du travail informatique. Le travail informatique consiste au développement de logiciels, c'est-à-dire, la conception de programmes en tant que produit final, où sont en rapport équipes, projets et réseaux. Le procès de travail connait d'importantes transformations en ce qui concerne les technologies de l'information qui donne lieu à un nouveau type de citoyenneté numérique, cependant l'exploitation est loin d'avoir disparu, s'adaptant aux nouvelles circonstances.

Logiciels libres et le travail informatique dans le capitalisme de la connaissance

Nous nous proposons dans ce travail d'explorer les relations entre les travailleurs informatiques dans le domaine de la production de logiciels et des services informatiques, à partir de la comparaison entre les modèles de développement de logiciels, le premier étant inspiré du logiciel libre, le deuxième des logiciels propriétaires.

Jusqu'aux années 80, la révolution informatique était une réalité visible, essentiellement aux États-Unis, en ce qui concerne les universités et les laboratoires. Les années 90, furent las années de la révolution informatique dans le monde entier, accélérée par la consolidation d'Internet.

Les théories sur l'économie de l'information ainsi que l'économie de la connaissance se mettent à jour, en conséquence.

Les communautés du logiciel libre, sont des communautés d'usagers innovateurs. Alors que le modèle de Microsoft du logiciel propriétaire se base sur les « nouvelles enclosures ». Le modèle du logiciel libre est un modèle « technoinstitutionnel » ouvert et coopératif, caractéristique des nouveaux biens publiques relatifs à l'information (Jollivet, 2007). Nous analyserons par la suite les deux modèles évoqués. Celui du logiciel libre nous intéresse afin de dévoiler certaines prises de positions dans le champ, qui reprennent les catégories de l'anthropologie économique comme moyen de légitimation. Dans le cas du

logiciel propriétaire, nous pointerons les inconsistances de certaines théorisations qui tendent à idéaliser les nouveaux types de travail. Nous adopterons également une perspective ethnographique pour analyser les pratiques et sens que ces derniers revêtent pour les travailleurs.

Le mouvement du logiciel libre

Depuis les années 80, de nombreux travaux ont été publié concernant le logiciel libre, notamment par Richard Stallman, fondateur de la Free Software Foundation. Une autre figure également importante pour le mouvement, est Eric Raymond, le fondateur d'Open Source Iniciative. Ses textes tels que A la conquête de la noosphère et La cathédrale et le Bazar, sont fondamentaux. Sans démontrer quelconque intérêt pour l'anthropologie, ni prétendre réaliser une ethnographie, l'auteur décrit les relations des travailleurs du secteur du développement du logiciel libre, qu'il qualifie d'économie du don. Malgré les limites évidentes, nous essayerons de pointer dans quelle mesure ces dernières peuvent nous illuminer sur des relations similaires dans la production de logiciel, insérée dans un procès de travail capitaliste.

Richard Stallman, fondateur en 1985 de la Free Software Foundation, a créé une communauté de production et d'échange de logiciels basée sur la coopération. Rappelons que dans le contexte nord-américain fortement favorable aux brevets, il était question de transformer tout les codes source ouverts, en code source propriétaires. Aussi, il a développé un système d'exploitation qui diffère d'Unix (même s'il est basé sur ce dernier, selon Stallman, GNU signifie « GNU n'est pas Unix »). Précisons que le mouvement du logiciel libre nait dans le but de s'opposer à la commercialisation du logiciel en général.

Le mouvement prend une ampleur considérable quand Linus Torvalds, étudiant finlandais, écrit un « kernel », ou noyau du système d'exploitation, compatible avec les « kernels » Unix du logiciel propriétaire. Torvalds a su combler les carences du projet GNU, désormais GNU/Linux, qui n'étant plus l'apanage des experts, attire de la sorte, l'attention des grandes compagnies d'informatique telles que IBM, HP, Sun Microsistems ou Apple.

Dans le domaine du développement de logiciel, nous pouvons distinguer deux courants opposés, celui du « logiciel libre » et celui de l'initiative d'Open Source, représentée par Eric Raymond, plus intéressé par les bénéfices technologiques et économiques qui découlent du code source ouvert. Open Source Iniciative, créée par Eric Raymond, contrairement à FSF, permet à des tiers la modification et la redistribution du logiciel, sous la forme du logiciel propriétaire.

La position de Raymond, remet en question le logiciel propriétaire, néanmoins les raisons diffèrent de celles de Stallman. Celui-ci ne s'oppose pas en principe aux sociétés de l'industrie informatique. Selon Raymond, le logiciel propriétaire représente une entrave à l'innovation technologique, car il empêche le développement de logiciels de meilleure qualité. Stallman, en revanche, défend

avant tout la liberté des usagers de comprendre le fonctionnement des logiciels, au détriment de leur efficience.

Par conséquent, Raymond réalise une description des tâches des programmeurs de logiciels et propose une théorisation qui justifie le développement créatif du logiciel, indépendamment du fait qu'il soit libre ou propriétaire. Ne faisant pas partie du monde académique, il emploie néanmoins des catégories de l'anthropologie économique de façon non spécialiste, afin de conformer les pratiques au sein du mouvement.

Les usages des catégories de l'anthropologie économique dans l'étude du mouvement du logiciel libre

Construire une communauté

Pour Raymond, amorcer un projet de logiciel libre équivaut à constituer une communauté, il recommande donc de suivre ce qu'il nomme « Le modèle du bazar ». Ce dernier fait allusion à l'étude de Cleeford Geertz au Maroc, qui décrit une « économie de bazar » où la négociation sur un marché houleux d'objets et de pratiques substitue l'imposition par la force. Raymond propose un schéma qui a inspiré et inspire toujours la production de logiciel. Il prétend rendre explicite la logique de fonctionnement, à première vue anarchique, de la production de logiciel, mais non pas dans le seul but de révéler une dynamique inconsciente. Il est question de mettre en relief la dimension instrumentale qui y apparaît afin d'améliorer la production.

« L'histoire d'Unix aurait dû nous préparer à ce que nous découvrons avec Linux (et à ce que j'ai expérimenté à une échelle plus modeste en copiant délibérément les méthodes de Linus): alors que l'acte de programmation est essentiellement solitaire, les plus grandes bidouilles proviennent de la mise à contribution de l'attention et de la puissance de réflexion de communautés entières. Le développeur qui n'utilise que son propre cerveau dans un projet fermé ne tiendra pas la route face au développeur qui sait comment créer un contexte ouvert, susceptible d'évoluer, dans lequel la traque des bogues et les améliorations sont effectuées par des centaines de gens. » (Raymond, 1997: 25).

Pour certains programmeurs, la production d'un logiciel comportait la même logique que celle de la construction d'un immeuble. La construction s'apparenterait à celle d'une cathédrale, une planification minutieuse qui prend en compte chaque étape d'un projet qui implique un grand nombre de variables afin d'éviter son effondrement. Cependant, au regard de son expérience personnelle, Raymond remarqua que la majorité des projets réussis ne suivaient pas ce cheminement. Il était plutôt question de projets où les règles n'étaient pas formellement établies, mais qui certainement appelés à un esprit de collaboration et à un type de travail qui ne correspondait pas à une planification anticipée pour chacun des éléments constitutifs. Le réseau de collaboration est devenue alors la règle générale de tous les projets, provoquant de la sorte, l'émergence plus ou moins spontanée d'une communauté de programmeurs.

Raymond propose alors, de manière explicite, que la création exclusive d'une telle communauté soit la condition nécessaire au succès du « modèle du bazar » : « Il est assez évident qu'il n'est pas possible de commencer à coder dans le style bazar dès le début. On peut tester, déboguer et améliorer un programme dans le style bazar, mais il sera très difficile de un projet dans le mode bazar. Linus ne l'a pas tenté, moi non plus. Il faut que votre communauté naissante de développeurs ait quelque chose qui tourne, qu'on puisse tester, avec quoi elle puisse jouer. Quand vous initiez un travail de développement en communauté, il vous faut être capable de présenter une *promesse plausible*. Votre programme ne doit pas nécessairement fonctionner très bien. Il peut être grossier, bogué, incomplet, et mal documenté. Mais il ne doit pas manquer de convaincre des co-développeurs potentiels qu'il peut évoluer en quelque chose de vraiment bien dans un futur pas trop lointain.» (Raymond, 1997: 21).

Les caractéristiques du responsable de projet, ses qualifications ainsi que le statut des codes doivent pouvoir être évalués par les éventuels membres de la communauté avant de prendre la décision d'en faire partie. Signalons toutefois, que le fait d'appartenir à une communauté ne s'acquiert pas de manière directe ou automatique.

La communauté des hackers, nous dit Raymond, exige pour intégrer ses membres qu'ils comprennent le « jeu des réputations » :

« Certains pourraient, incidemment, soutenir le fait que la structure même de la culture du don des hackers est son propre mystère. On n'est pas considéré comme acculturé (concrètement, personne ne l'appellera un hacker) avant d'avoir démontré un bon niveau de compréhension du jeu des réputations et de ce qu'il implique: coutumes, tabous, et usages. Mais c'est évident : toutes les cultures exigent une telle compréhension de la part de ceux qui manifestent la volonté d'entrer. De plus, la culture hacker ne manifeste aucune envie de voir sa logique interne gardée secrète — ou, au moins, personne ne m'a jamais incendié pour l'avoir révélée ! »

Culture hacker comme culture du don

Nous exposerons à présent les raisons pour lesquelles Raymond avance que nous nous trouvons face à une culture du don. L'auteur soutient que « Pour comprendre le rôle de la réputation dans la culture du code ouvert, il est utile d'aller chercher dans l'histoire de l'anthropologie et de l'économie, afin d'examiner la différence entre les cultures d'échange et les cultures du don. » (Raymond, 1999 : 9).

Pour l'auteur, la culture du don constitue un modèle qui aurait surpasser le modèle du commandement hiérarchique, ainsi que le modèle de l'échange. En ce qui concerne le premier modèle, la place occupée par les biens rares est décidée par une autorité centrale, soutenue par la force. Notre société correspond le plus souvent au second type, essentiellement une culture de l'échange, ce qui implique que « La plupart des gens ont un modèle mental implicite pour les deux systèmes décrits précédemment, et sur la manière dont ils interagissent. Le gouvernement, l'armée, et le crime organisé (par exemple)

sont des pouvoirs centralisés qui parasitent l'économie d'échange, plus vaste, que nous appelons le « marché libre ». Il existe cependant un troisième modèle, radicalement différent des autres et rarement reconnu en tant que tel sauf par les anthropologues ; la culture du don. »

Les caractéristiques que Raymond attribue à la culture du don sont les suivantes : « Les cultures du don ne sont pas des réponses à une pénurie, mais à une abondance. Elles surviennent dans des populations qui ne souffrent pas de carences significatives en biens de première nécessité. On peut observer des cultures du don en action dans les cultures aborigènes vivant dans des éco-zones au climat doux et à la nourriture abondante. On peut aussi les observer dans certaines strates de notre propre société, particulièrement dans le monde du spectacle et chez les gens très riches." (Raymond, 1999: 9). Selon Raymond, à la différence des sociétés primitives où le don est analyser, pour la culture hacker le prestige est le moteur de la compétition dans un contexte d'abondance et non pas de pénurie : « L'abondance rend les ordres imposés par la force difficiles à justifier et les échanges commerciaux presque sans objet. Dans une culture du don, le statut social n'est pas déterminé par ce que vous contrôlez, mais par ce que vous donnez. » (Raymond, 1999: 10)

La «compétition pour le prestige»

Les principales causes qui sous-tendent la recherche du prestige renvoient tantôt à certaines caractéristiques générales comme à certaines particularités de la culture hacker, qui la positionne comme le seul moyen d'obtenir un « statut ». Tout d'abord, car pour les individus, il s'agit d'une « récompense primaire » : « Deuxièmement, le prestige est un bon moyen (et dans une pure culture du don, *l´unique* moyen) d'attirer l'attention et la coopération des autres. Si quelqu'un est bien connu pour sa générosité, son intelligence, son honnêteté, son charisme, ou pour d'autres qualités, il lui devient bien plus facile de convaincre les autres qu'ils gagneront à s'associer avec lui. Troisièmement, si votre économie du don est en contact ou mélangée avec une économie d'échanges ou un pouvoir centralisé, votre réputation peut déborder de votre milieu original et vous faire atteindre dans le deuxième un statut plus élevé. (Raymond, 1999: 11).

Au-delà de ces raisons générales, les caractéristiques particulières de la culture des hackers font que le prestige est encore plus précieux qu'il ne le serait dans une culture du don du « monde réel ». La principale de ces « conditions particulières » est que les artefacts qui sont donnés à la communauté (ou, si on l'interprète différemment, qui représentent le signe visible de l'énergie et du temps déployés) sont très complexes. Leur valeur n'est pas aussi évidente que celle de dons matériels ou de l'argent dans une économie d'échanges. Il est plus difficile de distinguer objectivement un don exceptionnel d'un don médiocre. Par conséquent, la réussite de l'enchère de qui brigue un statut social plus élevé dépend fortement du jugement critique de ses pairs. Une autre particularité de la culture des logiciels à source ouverte est sa relative pureté. La plupart des cultures du don sont compromises — soit par des liens avec l'économie d'échange tel que le commerce de biens de luxe, soit par des

relations avec un pouvoir centralisé tels que des regroupements en familles ou en clans. Il n'existe rien de tel dans la culture des logiciels à source ouverte. En dehors de la réputation aux yeux de ses pairs, il n'y a virtuellement aucun salut. (Raymond, 1999: 11-12).

La propriété concernant le projet du logiciel libre

La reconnaissance du droit à distribuer le logiciel et ses versions modifiées par la communauté, associe la propriété à son développement. Cette propriété d'un type particulier de biens, qui ne sont que des idées, des codes, des algorithmes, etc...et qui ne se trouvent physiquement en aucun lieu.

Malgré le fait que la propriété du projet peut être sujette à modification, les mécanismes pour y aboutir sont loin d'être naturels. Si le fondateur du projet y perd son intérêt ou ne réalise pas le support technique nécessaire et que d'autres éditeurs veulent le faire, la succession de la propriété fait l'objet de discussions ardues.

Pour Raymond, il ne faut pas confondre « noosphère » avec univers virtuel. La « noosphère » est une sorte de réservoir d'idées alors que l'univers virtuel est un espace virtuel dans lequel se trouvent les idées. Le développement du logiciel libre implique le fait de « cultiver la noosphère ». Les discussions concernant la possibilité de breveter le logiciel , lui assigner des droits de propriété intellectuelle est une thématique qui traverse toutes ces activités. Raymond se positionne en tant que défenseur de la Open Source Iniciative, qui cherche à affranchir les contenus de l'univers virtuel des encadrements qu'imposent les droits de propriété, néanmoins il défend également, l'idée de propriété pour le champ de la « noosphère », qu'il met en relation avec l'éthologie. Il affirme donc que: « Clamer sa propriété (ainsi que définir son territoire) est un acte performant, c'est un moyen de déclarer quelles frontières seront défendues. Appuyer ce droit à la propriété est un moyen de minimiser les conflits et de maximiser un comportement coopératif. Cela reste encore une réalité lorsque le « droit à la propriété » est plus abstrait qu'un enclos ou qu'un aboiement de chien, alors même qu'il est réduit à l'apparition du nom d'un responsable de projet dans un fichier. Il s'agit toujours d'une abstraction de la territorialité, et (comme dans d'autres formes de propriété) nos modèles instinctifs de propriété sont des modèles territoriaux qui ont évolué en vue de résoudre les conflits.»

Les pratiques et les tabous

Programmer, « picar codigo », selon les éditeurs, constitue leurs activité principale. La possibilité de modifier le code et de distribuer les programmes font partie des raisons pour lesquelles ils défendent le logiciel à source ouverte. Cependant, Raymond distingue des tendances divergentes dans la culture hackers : « Tous ses membres s'accordent sur le fait que les logiciels à source ouverte (c'est-à-dire les logiciels librement redistribuables et qu'on peut facilement faire évoluer et modifier en fonction de ses besoins) sont une bonne

chose et valent la peine de s'y consacrer de façon significative. Cette position définit efficacement l'appartenance à cette culture. Pourtant, les raisons pour lesquelles des individus et différentes sous-cultures adhèrent à cette croyance sont très variées.” (Raymond, 1999: 1).

Raymond établie, de la sorte, une distinction entre les « idéalistes » et les « pragmatistes» au sein de la communauté. Les premiers sont représentés par la prise de position anti-commerciale et contre les entreprises. Les seconds ont une posture moins radicale : « Pour le pragmatiste, la est un outils important, mais pas une fin en soi. Son utilité principale n'est pas d'être une arme contre la rétention d'informations, mais plutôt un moyen d'encourager la communauté des développeurs au partage des logiciels et à l'adoption du modèle de programmation de type bazar. Le pragmatiste accorde plus de valeur aux bons outils et aux gadgets, il ne déteste pas le commerce et il peut utiliser des outils commerciaux de bonne qualité sans se poser de problème de conscience. En même temps, son expérience des logiciels à source ouverte l'a habitué à une qualité technique que très peu de logiciels fermés peuvent atteindre. » (Raymond, 1999: 3).

Le développement du système Linux, aurait affecté la position des « idéalistes », selon Raymond : « De plus en plus, les anti-commerciaux fanatiques se retrouvèrent en minorité. Le fait que les choses aient changé ne devint apparent qu'à l'annonce faite par *Netscape* en février 1998 de la mise en libre accès des sources de *Navigator 5.0*, ce qui a attisé l'intérêt que l'industrie portait au « logiciel libre ». Cette annonce sans précédent a eu pour conséquence le changement de nom de « logiciel libre » à « logiciel à source ouverte », qui s'est produit avec une facilité qui a surpris tous les acteurs de cette affaire.” (Raymond, 1999: 3).

Lorsque Raymond analyse les pratiques d'informaticiens, il souligne que ces derniers répondent à une « théorie de la promiscuité » du mouvement du logiciel libre qui s'oppose à une « pratique puritaine ». En effet, l'existence de certains tabous qui nous confirment indirectement la fonctionnalité de la théorie héritée de Locke sur la propriété, sur le plan matériel ainsi que sur le champ virtuel. Pour Raymond, les pratiques des informaticiens dans la création de logiciels renvoient à la théorie de la propriété de Locke, et les coutumes relatives à la propriété sont « un moyen de maximiser les incitations à la réputation » (Raymond, 1999: 12).

A priori, n'importe quelle personne serait apte à opérer en tant que hacker ou sur n'importe quel produit à code source ouvert. Toutefois, Raymond nous fais remarquer que : “En pratique, pourtant, de telles « scissions » n'arrivent quasiment jamais. Les ruptures dans les projets majeurs ont été rares, et toujours accompagnés d'un changement de nom et d'un grand nombre de justifications publiques. Il est clair dans des cas comme la séparation de *GNU Emacs/XEmacs*, celle de *gcc/egcs*, ou encore les différents schismes des groupes de, que les dissidents sentaient qu'ils s'opposaient à une norme partagée par tous.” (Raymond, 1999: 3).

Les trois tabous que Raymond signale comme propres à la culteure hacker confirment ce qui précède :

1) Il existe une forte pression sociale contre la scission de projets. Cela n'arrive pas, sauf si l'on plaide l'absolue nécessité avec de nombreuses justifications publiques et un changement de nom du projet. Le risque encouru est de perdre en réputation « ils ne peuvent le contrôler qu'en étant actifs simultanément dans les projets après la scission » (Ceci prête généralement à confusion et serait difficile à mettre en œuvre en pratique)

2) Distribuer des modifications à un projet sans la coopération des modérateurs est mal vu, sauf dans certains cas, comme par exemple des corrections mineures pour porter le logiciel sous une nouvelle architecture.

3) Enlever le nom d'une personne de l'histoire d'un projet, des remerciements ou de la liste des mainteneurs est *Absolument impossible* sans le consentement explicite de la personne.

D'après Raymond, malgré les déclarations publiques des hackers qui remettent en question l'égoïsme et les motivations impulsées par l'égo, tout en réalisant un culte à l'humilité, masquant de la sorte leur propensions à l'honneur et au prestige individuel. Reconnaître cela conduirait à corrompre les bases de l'esprit créatif et coopératif. Par conséquent, l'humilité tourne à l'avantage d'une productivité croissante.

Comment est valorisé un don ou un cadeau

Raymond relève qu'il existe des règles qui encadrent la valorisation des solutions informatiques apportées dans la communauté. Premièrement, que le programme fonctionne selon les expectatives, même s'il est toléré qu'il y est des bogues ou des erreurs. Deuxièmement, il est plus valoriser d'étendre la « noosphère » plutôt que de dupliquer une pièce déjà existante, à l'exception des logiciels à source fermée. Troisièmement, la production qui est distribuée, c'est-à-dire en circulation, est également plus valorisée. « Les distributions principales englobent non seulement les grandes distributions de Linux comme Red Hat, Debian, Caldera et S.U.S.E, mais aussi d'autres distributions qui se maintiennent et se certifient elles-mêmes de par leur réputation. (telles que les distributions BSD ou la série de sources de la Free Software Foundation) » Quatrièmement, le travail à l'usage de tous est considéré meilleur, car cela signifie que les solutions alternatives ne fonctionnent pas aussi bien. Enfin, « l'entière dévotion pour le travail difficile et fastidieux (comme le débogage ou écrire de la documentation) est plus prestigieux que le travail considéré amusant et facile. » (Raymond, 1999: 18).

Propriété, structure du projet et résolution des conflits

Pour éviter ces conflits, Raymond propose une structure de projets fondée sur le modèle du « dictateur bénévole », qui comme nous le verrons ultérieurement, est aussi le modèle prédominant dans la production de logiciel propriétaire :

« Le cas non trivial le plus simple est lorsqu'un projet a plusieurs co-mainteneurs qui travaillent sous la direction d'un « dictateur bienveillant ». L'usage favorise ce type d'organisation pour les projets de groupes. L'expérience montre que pour des projets aussi gros que le noyau *Linux* ou *Emacs* cela fonctionne correctement, et résout le problème du « Qui décide » d'une façon qui n'est pas forcément la pire de toutes les possibilités." (Raymond, 1999: 21).

Le dictateur est supposé consulter les co-mainteneurs dans la prise de décision : « Quand les projets de type dictateur-bienveillant rassemblent plus de participants, deux familles de contributeurs tendent à se démarquer : les contributeurs ordinaires et les co-développeurs. Un cheminement typique pour devenir un co-développeur est de prendre la responsabilité d'un sous-système majeur du projet. Un autre est de se transformer en « Chasseur de bogues », en débusquant et en corrigeant un grand nombre de bogues. De cette façon ou d'une autre, les co-développeurs sont des contributeurs qui s'investissent de façon substantielle et persistante dans le projet. » (Raymond, 1999: 22).

Productivité dans le projet du logiciel libre

A la suite de son argumentation, émerge le fondement de la thèse soutenue par Raymond : « Le verdict de l'histoire semble être que le capitalisme et le libre marché est une façon globalement optimale de coopérer pour engendrer une économie efficace. Peut-être que, d'une manière similaire, le jeu des réputations de la culture du don est la façon globalement optimale de coopérer pour créer (et contrôler !) un travail créatif de qualité. » Les raisons avancées pour soutenir le logiciel non propriétaire sont liées à la productivité, non pas à la contestation d'un système qui se maintient par la privatisation d'une collaboration commune. « Un groupe de développeurs de codes source ouverts sera considérablement plus productif (particulièrement à long terme, car la créativité devient plus critique comme un multiplicateur de productivité), alors qu'un groupe de même taille, connaissances, de programmeurs de codes source fermés sont (dé)motivés par les récompenses ». (Raymond, 1999: 25).

Selon l'auteur, ceci est le résultat des intentions propres des programmeurs. «La ``fonction d'utilité'' que les bidouilleurs Linux maximisent n'est pas classiquement économique, c'est l'intangible de leur propre satisfaction personnelle et leur réputation au sein des autres bidouilleurs. (On peut être tenté de penser que leur motivation est ``altruiste'', mais c'est compter sans le fait que l'altruisme est en soi une forme d'égoïsme pour l'altruiste). » L'individualisme serait donc toujours présent, favorisé et accrue par la communauté : « Je pense qu'à l'avenir, le logiciel dont le code source est ouvert sera de plus en plus entre les mains de gens qui savent jouer au jeu de Linus, des gens qui abandonnent les cathédrales pour se consacrer entièrement au

bazar. Cela ne veut pas dire que les coups de génie individuels ne compteront plus ; je pense plutôt que l'état de l'art du logiciel dont le code source est ouvert appartiendra à ceux qui commencent par un projet individuel génial, et qui l'amplifieront à travers la construction efficace de communautés d'intérêt volontaires. » (Raymond, 1999: 26).

Quelques considérations sur le travail de Raymond

A la suite de ce parcours à travers l'œuvre d'Eric Raymond, nous pouvons souligner l'usage des catégories de l'anthropologie économique pour rendre compte des pratiques dans la production de logiciels, mais aussi pour les légitimer. Telle est la perception d'un des membres les plus proéminent de la communauté des logiciels libres. La proximité de l'analyse de Raymond avec les théories libérales qui défendent la compétition (Adam Smith) et la propriété (John Locke) mérite de nous interpeller afin de prêter une attention particulière aux usages abusifs de l'idée de don. L'anthropologie économique s'est constituée justement en s'opposant à ces visions réductionnistes de la complexité du social. Raymond s'approprie des catégories pour les réemployer de manière décontextualisée, qui n'interroge pas le point de vu des sujets impliqués.

Notre regard s'est porté sur le fait que l'auteur s'en remette au courant de pensée économique néoclassique pour faire référence aux conduites des développeurs. L'Ecole néoclassique défend l'existence d'agents rationnels et atomistiques qui maximisent des fonctions objectifs, « l'utilité », selon les dotations dont ils disposent, indépendamment de leur contexte social ou culturel, ce qui n'entre pas en contradiction avec les considérations d'un individu dont les pratiques sont immerger dans toute une série de conditionnements et significations sociaux. Si Polanyi, en référence à l'économie classique de Smith et Ricardo, remettait en question « l'émergence du credo libéral » et la notion de « marché autorégulé », de nos jours nous ne pouvons que contester la référence faites aux économistes néoclassiques pour caractériser l'interaction des individus dans et hors du champ économique.

Ainsi, l'auteur nous renvoie à un champ théorique libéral pour justifier l'appropriation des biens de la « technosphère ». La référence au libéralisme classique repose sur la pensée de Locke concernant la défense du droit de propriété sur la base du travail. Nonobstant, en philosophie politique la tradition contractualiste naturalise la propriété en tant que droit individuel. Ceci s'oppose à l'idée selon laquelle la création d'idées et de connaissances provient de ce qui est « commun ». La notion marxiste de Generall Intellect nous paraît bien plus appropriée pour analyser ces nouvelles dimensions de de l'appropriation (Benkler, Negri). Nous proposons, par conséquent, une approche au champ économique, à travers la critique de l'Economie Politique commune à de nouveaux courants, qui proviennent du marxisme, remarquables de par leur pouvoir d'explication concernant les caractéristiques du travail au XXI siècle.

Le travail de Raymond n'invalide pas, néanmoins, la possibilité d'approcher les pratiques des développeurs de logiciels libres dans une perspective ethnographique, qui peut être de grande utilité pour aborder ce type de relation. Il nous semble intéressant de transposer ces considérations sur la pertinence d'une approche ethnographique, à l'étude du procès de travail informatique « propriétaire », non libre. Il faudrait toutefois, envisager la possibilité de reprendre à notre compte les leçons de la sociologie du travail et de la critique de l'économie politique, dans un dialogue qui semble potentiellement fructueux. Dans le présent travail, je signale l'intérêt de construire des ponts dans cette direction, ce qui renforce la conception de Florence Weber, selon laquelle l'anthropologie économique, loin d'être confinée aux sociétés primitives, traditionnelles, non occidentales, peut rendre compte de la complexité qu'englobe le travail informatique, élément central des nouvelles technologies de la dernière phase du capitalisme avancé. (Weber, 2009: 29-34).

Organisation du travail informatique : équipes, projets et réseaux

Le travail informatique consiste à développer le logiciel, c'est-à-dire, l'élaboration de programmes comme produit final. La singularité du produit, qui le différencie des marchandises traditionnelles, est son caractère immatériel. Cependant, ceci n'empêche pas que comme les produit traditionnels offerts par l'industrie, ils soient l'objet d'études à partir des étapes que comprennent son « cycle de vie », expression très rependue dans le domaine de l'informatique. Ces étapes sont, d'une part : la prise en considération des conditions requises par le client, la conception, l'architecture du logiciel, l'analyse fonctionnelle, les mises à l'épreuve partielles et d'ensemble, l'application et l'entretien. (Castillo, 2009). D'une autre part, une autre caractéristique fondamentale, est le travail en équipe, en relation avec de réseaux et en fonction de différents projets, la coopération en réseau devient donc centrale.

La coordination de différentes « ressources » dans la quête d'un objectif particulier et pour un temps limité, justifie l'apparition de la figure des « responsables de projets ». La figure du chef de projet revêtit des caractéristiques du « manager », qui donne des ordres et attend les consignes de la direction, c'est « l'homme des réseaux ».

L'organisation « horizontale » du travail

Le travail dans la programmation informatique se présente comme exemple d'une organisation moins hiérarchique, où circule la connaissance, ceci est généralement imputé à la nature même du produit. En général, écrire des programmes signifie résoudre toute une série de « nouveaux » problèmes, ordonner des valeurs, les disposer dans des structures efficientes de mémoire, livrer des interfaces d'usagers claires, équilibrer le poids de l'ensemble des processeurs, échanger des données entre ordinateurs, lire et écrire différents formats de données, entre autres.

Pour Martin, qui fait de la programmation depuis ses 15 ans, programmer c'est “picar codigos”, ce qui requiert chaque fois plus d'exigence, cependant, mettre en jeu chaque fois plus de capacités n'implique pas acquérir plus de droits sur le produit de son travail :

Martin : *La différence c'est principalement les outils qu'on a en tant que développeur. Aujourd'hui ce sont des outils plus agiles, ils sont beaucoup plus complexes, ils permettent attaquer...faire des programmes complexes. Un programme complexe ne peut pas avoir une solution simple, c'est-à-dire, avec les outils dont on dispose aujourd'hui. Avant, il y avait un autre type de programmation, ce n'était pas de la programmation orientée à des objets, par exemple...*

Pablo : *Aujourd'hui pour programmer, un développeur a plus d'exigence?.*

Martin : *Beaucoup plus d'exigence.*

Pablo: *Et cela va toujours en crescendo, ça ne peut pas être autrement?.*

Martin: *Les technologies évoluent.*

Pablo: *Et le temps de formation? C'est plus facile d'apprendre?.*

Martin: *Non, non, pas du tout...*

Pablo: *Comment ça se comptabilise...vous étudiez plus qu'avant?.*

Martin: *Ce qui se passe, c'est que maintenant il y a plus d'information, il y a plus de moyens ou de chemins pour que celui qui veuille être développeur le fasse par ses propres moyens, pour réduire la courbe de l'apprentissage, les entreprises mettent des personnes disponibles pour la formation basique, de ses méthodes de travail, des outils qu'il a déjà préparé pour être réutilisé. C'est comme avoir des pièces ou des boîtes noires ou des systèmes. Il faut utiliser ce qui est déjà constitué, qui est garantit, qui a de la qualité, que nous savons qui marche.*

Comme n'importe quel autre travail, Martin met à disposition de l'employeur sa « capacité logique » et il ne considère pas que le produit de son travail doive lui appartenir :

Martin: *Tu travailles pour ton employeur...toutes les lignes de codes qui sont écrites ne t'appartiennent pas...toi, juste t'es en train de vendre tes connaissances et tu les exprimes sous forme de logiciel...ta « capacité logique »...pour créer des lignes de codes ...si c'est pas toi qui le fait, ça sera un autre qui le fera.*

Pablo: *Pourquoi? Je veux dire...si chacun est différent.*

Martin: *Chacun peut le faire de manière différente, pour que notre solution soit brillante et la solution des autres moins brillante.*

Pablo: *Et ça ne t'affectes jamais de savoir que t'as trouvé la solution du problème et qu'elle n'est pas à toi?.*

Martin: *Non, parce que justement un développeur se consacre à répondre à ce que lui demande son employeur.*

Défi, roulement et leadership : les nouvelles formes d'exploitations (et les anciennes)

Dans ce secteur, le roulement dans les niveaux inférieurs (junior) de développement est fréquent, ce qui s'oppose au supposé « engagement » promue par les chefs de projets. Pour Daniel, chef d'une petite entreprise de développement de moins de 10 employés, la division du travail suppose embaucher des travailleurs formés avec des capacités qui vont au-delà de l'aspect technique:

Daniel: *D'abord il y a l'analyse, recensement, enquêtes, outils pour faire une bonne analyse, problématiques associées. Ensuite, il y a la conception, et cette partie est plus coûteuse, beaucoup plus coûteuse, elle requiert plus de ressources, de meilleurs ressources.*

Pablo: *C'est quoi les ressources?.*

Daniel: *J'ai besoin de plus de programmeurs, de bons analystes-programmeurs qui aient un bon feeling avec le client, je fréquente les gens, parce que ce sont eux qui sont avec la personne qui a le problème.*

Cependant, il considère que le roulement est inévitable et que le leadership défini le succès du projet:

Daniel: *Ça a beaucoup à voir avec les équipes de travail, avec les leaders.*

Pablo: *Les leaders du projet?.*

Daniel: *Oui. Les leaders du projet...Tous le considèrent comme un leader, alors il faut travailler sur ce point pour maintenir motivées les équipes de travail, afin de les faire perdurer.*

Pablo: *Comment travaille quelqu'un qui est jeune, qui est investit en tant que leader et qui est avec quelqu'un de plus grand, qui est là depuis plus longtemps...?*

Daniel: *Le leadership se développe en général dans ce domaine selon la plus grande connaissance. Quelqu'un qui est leader a des connaissances et de l'autorité parce qu'il a plus de connaissances qu'un autre.*

Pour Daniel, afin d'éviter le roulement, la tâche d'un bon leader est de proposer des « défis » à la hauteur du travailleur.

Pablo: *C'est-à-dire que le développeur est plus facilement remplaçable que l'analyste-programmeur?*

Daniel: *Oui, absolument. Si la partie du développement est bien organisée dans une entreprise qui se consacre au développement de logiciels...les développeurs sont ceux qui sont le plus l'objet de roulement.*

Pablo: *Pourquoi ils sont l'objet de roulement? Parce qu'on leur offre d'autres emplois?.*

Daniel: *Pour beaucoup de raisons, une des raisons c'est parce qu'ils n'ont pas la motivation suffisante par rapport à leur travail, par rapport à la qualité du travail qu'ils sont en train de faire. Ça ça pourrait être une raison...Ils veulent faire d'autres choses, utiliser d'autres langages, ils veulent apprendre d'autres choses. S'ils n'ont pas de défis intéressants, par exemple, s'ils ne doivent pas*

faire des programmes pour les banques, des entreprises multinationales, c'est-à-dire, des défis intéressant...les programmeurs partent...

D'après Daniel, le succès du projet dépend en grande partie du leader mais celui-ci doit sélectionner les travailleurs qui acceptent le défi de la tâche, mais qui ne se distraient pas en le réalisant. Pour Carlos, manager d'une entreprise de développement de moins de 10 salariés, le travail de ses employés est bien rémunéré, mais il reconnaît qu'il ne le ferait jamais.

Carlos: *C'est quelque chose qui s'apprend, un mec en trois ans il est en train de programmer et il gagne un salaire qu'un médecin...qu'il y a des médecins qui ne vont jamais le gagner... Tous les médecins ne gagnent pas 3000 balles par mois, sûrement... quelques uns en gagnent plus, mais en moyenne ils en gagnent moins...*

Pablo: *Mais c'est pas un peu fastidieux ce travail pour un jeune?.*

Carlos: *Moi je le ferais pas...même pas fou, celui de l'analyste-programmeur, bon, si on me paye trop bien, peut-être que je pourrais le faire...*

Pablo: *Pourquoi tu ne le ferais pas?.*

Carlos: *Parce que t'es assis en face de l'ordinateur tout le temps en train d'écrire quelque chose, tout le temps, je sais pas...t'es en train de faire quelque chose avec une seule interaction, d'ailleurs, moins tu interagis avec les gens pendant le procès de travail, mieux c'est, comme ça tu te déconcentres pas. Si le type est en train de penser pour écrire une ligne, il peut parler à personne, interagir avec quelqu'un pour faire son travail, après oui, dans une unité de développement. « regardes c'est ce que ça donne », mais une fois qu'on m'a donné les objectifs, je dois les écrire et les écrire le plus rapidement possible et les documenter bien...*

Pour Carlos, faire bien le travail- « picar codigo », dirait Martin- implique le moins de distraction possible et moindre interaction. Malgré le fait que le secteur est très dynamique, sur ce point il ne se distingue pas des autres travailleurs du travail industriel. Le roulement n'est pas lié à un manque de « défis », mais à l'excès de demande des travailleurs et les salaires du secteur relativement élevés. Pour Daniel, cependant, même s'il reconnaît que le roulement est un problème, il est plus important de trouver rapidement des remplaçants pour les programmeurs qui partent. Ici aussi, il propose des méthodes de gestion qui ne diffèrent pas des autres secteurs, qu'il soit ou pas dynamique au niveau technologique.

Pablo: *On est en train de parler d'une entreprise de 20, c'est une PME extrêmement petite, j'imagine que l'espace est un espace petit...un bureau.*

Daniel: *Oui, quand une ressource s'en va, l'impact est beaucoup plus grand...je sais pas, deux personnes par semestre partent...*

Pablo: *Comment tu planifie le travail en équipe à long terme, quand tu sais que ce sont des petites équipes avec beaucoup de roulement?.*

Daniel: *Et ça...chaque entreprise aura sa stratégie, ça fera partie du plan stratégique de chaque entreprise. Il faut miser beaucoup sur les réseaux, qui est-ce qu'on choisit comme leader d'un procès de développement, une*

méthodologie qui facilite le remplacement des développeurs, essayer de garder les développeurs junior, oui? Qu'ils soient facilement remplaçables parce qu'ils sont moins indispensables. Ça comment tu le fais? En faisant des outils, en utilisant des méthodes de développement et de travail en équipe où tu attribues peu de responsabilité aux développeurs junior, c'est plutôt des travaux automatisés ce que font les développeurs, tu leur enlèves le prise de décision.

Pablo: *Ça ça ne peut pas faire qu'effectivement ils s'ennuient, qu'ils perdent la motivation rapidement?.*

Daniel: *Mais sils continuent à travailler, ils vont grandir et ils vont réussir à être leader à un moment donné...*

Malgré le fait que la recherche ne soit pas achevée, à partir de quelques entretiens et d'observations sur le terrain, nous pensons que la logique propriétaire de la production de logiciel est assez cohérente avec la proposition de Raymond. La gestion des équipes de travail dans la production pour le marché révèle de grandes similitudes avec la forme de la communauté du logiciel libre décrite par Raymond. La compétition pour le prestige individuel se transforme, plus qu'en de meilleures salaires (relativement élevés), en un dépassement de soi-même et nouveaux défis. Le succès des projets renvoie aux conditions du leader et à son prestige fondé sur la connaissance. Toutefois, nous nous interrogeons sur la proposition de Raymond, si ce n'est pas une manière de légitimer une position qui fait passer avant l'efficience dans la production à la liberté de faire usage de la connaissance, que cela soit ou pas plus efficace.

La pertinence de l'approche ethnographique et le dialogue avec d'autres disciplines

A la lumière de ces observations, il nous semble que l'ethnographie représente une méthode appropriée pour rendre compte des pratiques au sein du mouvement du logiciel libre et dans la production du logiciel propriétaire. La perspective ethnographique ne se centre pas sur la technologie, mais sur les usages et les sens qui lui sont attribuées. (Hine : 2008). Les activités, les relations et les significations qui se forgent entre ceux qui participent des procès sociaux liés au monde du travail informatique, qui peuvent être tacites ou supposées, peuvent être explicitées. Les formes de la construction de sens dans le procès de travail informatique s'articulent directement avec les pratiques, mais pas exclusivement aux techniques associées aux nouvelles technologies. (Hine : 2008) Les assertions sur ces technologies présentent dans les médias, revues spécialisées et publications académiques ne pénètrent pas généralement dans les relations et significations complexes qui se tissent dans le procès de travail, où, en grande partie, le travail transcende le plan physique et où le produit est immatériel.

L'ethnographie apparaît alors comme une méthode particulièrement utile pour analyser les relations entre les travailleurs informatique, tant dans la production de logiciels libres que dans le champ des logiciels propriétaires. En faisant un usage non spécialiste des catégories de l'anthropologie économique, les textes

de Raymond vont dans cette direction, pour rendre compte du travail informatique, ainsi que pour légitimer ses positions au sein du mouvement.

Nous avons voulu à travers ce travail signifier les potentiels ponts que l'on peut établir entre l'ethnographie économique, la sociologie du travail et la Critique de l'Economie Politique, dans un dialogue qui se montre potentiellement bénéfique. Arturo Escobar s'interrogeait en 1994 « Serait-il possible de rendre compte à travers l'ethnographie de la multiplicité des pratiques associées avec les nouvelles technologies, dans divers contextes sociaux, ethniques et géographiques? De quelle manière ces pratiques sont liées à des thématiques sociales plus vastes comme par exemple, le contrôle de la main d'œuvre, l'accumulation du capital, l'organisation des modes de vie et la mondialisation de la production culturelle? » (Escobar 1994 : 22). Les catégories que les travailleurs informatiques mettent en jeu et mobilisent dans leurs pratiques établissent un dialogue entre l'ethnographie et les études du travail, et- en tant que travailleurs, acteurs économiques fondamentaux- avec l'économie. (Dufy et Weber, 2009 : 12)

Bibliographie

(2003) BENKLER, Yochai, “La Economía Política del procomún” en NOVATICA/UPGRADE, mayo/junio de 2003, nº 163, pp. 6-9, ATI, Madrid

(2009) CASTILLO, Juan José, “Las fábricas de software en España: organización y división del trabajo. El trabajo fluido en la sociedad de la información” en *Trabajo y Sociedad*, Nº 12, Vol XI, Otoño 2009, Santiago del Estero, Argentina

(2004) DYER- WHITEFORD, Nick, “Sobre la contestación al capitalismo cognitivo. Composición de clase de la industria de los videojuegos y de los juegos de ordenador” en Moulier Boutang, Yann, Corsanni, Antonella, y Lazzarato, Maurizio y otros (2004): *Capitalismo cognitivo, propiedad intelectual y creación colectiva*, Traficantes de sueños, Madrid.

(2009) DUFY, Caroline y WEBER, Florence, *Más allá de la Gran División. Sociología, economía y etnografía*., Ediciones Antropofagia, Buenos Aires.

(2005) [1994] ESCOBAR, Arturo, “Bienvenidos a Cyberia.Notas para una antropología de la Cibercultura.”, *Revista de Estudios Sociales* Nº 22, diciembre de 2005, Universidad de Los Andes, Colombia, 15-35.

(2001) HINE, Christine, *Etnografía virtual*, Editorial UOC, Barcelona.

(2007) JOLLIVET, Pascal: “Los rendimientos crecientes de adopción creativa ¿Hacia una competencia entre dos modelos tecnoinstitucionales en el sector software?” en (2007) RIVERA RIOS, Miguel y DABAT, Alejandro (coords.): *Cambio histórico mundial, crecimiento y desarrollo.*, Universidad Nacional Autónoma de México, México DF.

(1997) RAYMOND, Eric: “La catedral y el bazar” ,Publicacion electrónica disponible en http://biblioweb.sindominio.net/telematica/catedral.html.

(1999) RAYMOND, Eric: “Cultivando la noosfera” Publicacion electrónica disponible en http://www.geocities.com/jagem/noosfera.html

(2004) VERCELLI, Ariel: *La conquista del Ciberespacio: Creative Commons y el diseño de entornos digitales como nuevo arte regulativo en Internet.*, Publicacion electrónica disponible en http://www.arielvercelli.org/lcsdc.pdf, Buenos Aires.

(2006)YOGUEL, Gabriel, ERBES, Analía, y ROBERT, Verónica: “El sendero evolutivo potencialidades del sector de software en Argentina” en (2006) BORELLO, José, ROBERT, Verónica y YOGUEL, Gabriel (comps.): *La informática en Argentina. Desafíos a la especialización y a la competitividad*., UNGS - Ed. Prometeo, Buenos Aires.