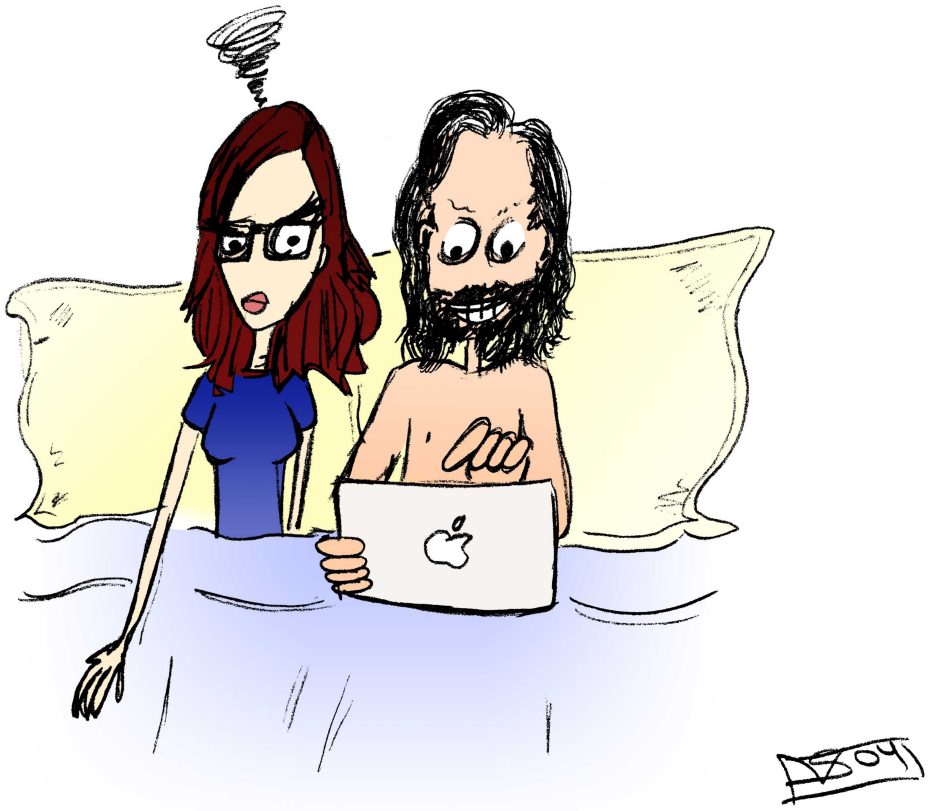
Étude du logiciel libre
Une innovation sociale



Mémoire présenté par Stéphane Rieppi en vue de l'obtention
du grade de licencié en sociologie

Institut des Sciences humaines et sociales

Promoteur : M. Jim Lindsey

Lecteurs : M. Christophe Lejeune

M. Michel Laffut

Année académique 2004 – 2005

Étude du logiciel libre
Une innovation sociale

Mémoire présenté par Stéphane Rieppi en vue de l'obtention
du grade de licencié en sociologie

Institut des Sciences humaines et sociales

Promoteur : M. Jim Lindsey

Lecteurs : M. Christophe Lejeune

M. Michel Laffut

Année académique 2004 – 2005

Je remercie mon promoteur et mes lecteurs pour leur soutien dans la réalisation de ce projet.
Je remercie particulièrement Christophe Lejeune pour ses conseils avisés et sa patience.
Rien n'aurait été possible sans le soutien de Stéphanie, que je remercie chaleureusement.
Merci aussi à Martin, pour nos longues discussions sur le logiciel libre.
Merci à tous ceux qui ont relu ce travail pour leurs remarques constructives.

Merci à ma famille et à mes proches.

Je dédie ce travail à la mémoire de René Doutrelepont,
qui avait accepté de m'accompagner dans ce projet.

Introduction et méthodologie

I. Choix de l'objet

Lorsque j'annonçai pour la première fois à mon entourage que j'allais travailler sur le logiciel libre, sa réaction fut quasi-unanime : sociologues comme profanes me regardaient, perplexes, me demandant si je ne perdais pas de vue que j'avais pour but de créer un texte à vocation sociologique. Pourtant, au fond de moi, je savais que ce terrain, certes original, ne manquerait pas de perspectives sociologiques. Il ne me restait plus qu'à arriver à verbaliser ma motivation, tâche plus ardue que je ne l'aurais cru de prime abord. Après quelques mois au cours desquels quelques bafouillements embarrassés étaient tout ce que je pouvais fournir pour toute réponse à cette terrible question – « *Es-tu sûr que tu fais de la sociologie ?* », je commençai à trouver une réponse qui, au fur et à mesure que je l'élaborais, rassurait chaque fois un peu plus mes interlocuteurs sur la pertinence sociologique de mon choix. Et me confortait dans mon choix.

Cette réponse, en définitive très simple est que, contre toute attente, **la principale innovation proposée par le logiciel libre est essentiellement sociale**. Au lieu d'un modèle de développement traditionnel, en entreprise, fortement centralisé et hiérarchisé, motivé par un salaire, nous avons un modèle de développement fortement décentralisé, *a priori* dépourvu de motivation pécuniaire. Cette motivation serait plutôt une **recherche d'utilité sociale** ou de **reconnaissance**, voire même, chez des auteurs comme Himanen, de créer une nouvelle éthique, un nouveau modèle de société. Himanen, dont je parlerai tout au long de ce texte, calque d'ailleurs son analyse sur celle de Max Weber et décrit la naissance d'une « ère de l'information », produite par l'*éthique hacker*, au même titre que l'éthique protestante avait engendré l'émergence du capitalisme chez Weber.

II. Le défi d'un terrain virtuel

Mon objet d'étude sera essentiellement virtuel et implique le défi de travailler sur un terrain qui l'est tout autant. Dès à présent, nous devons tenter non de définir mais tout au moins de baliser ce qu'implique ce terme « virtuel », qui, s'il a souvent été utilisé comme catégorie « fourre-tout » pour classer « ce qu'on ne comprend pas », n'en a pas moins, paradoxalement, une réalité qui lui est propre.

Travailler dans un domaine d'investigation échappant au tangible implique d'avoir recours à des outils propres à ce domaine. Il est toutefois évident que de grands rapports de parenté lient ces outils à leurs équivalents plus traditionnels. Une importante source d'inspiration à ma réflexion quant à l'étroitesse de ces rapports me fut donnée par le séminaire « *Le virtuel et le tangible : ce qui résiste* » organisé par l'Université Technologique de Compiègne, auquel j'ai assisté du 25 ou 28 janvier 2005.

Les communautés de développeurs que j'analyserai, seront virtuelles dans le sens où, la plupart du temps, ses membres ne se sont pas rencontrés « dans la vraie vie[1] », mais uniquement *via* les moyens que les nouvelles technologies de l'information mettent à leur disposition, à savoir principalement le courrier électronique, les forums de discussion, les systèmes de messagerie instantanée ou, plus classiquement, le téléphone et les dérivés de celui-ci (téléphonie par Internet, vidéo-conférence, etc.).

Ma démarche aura, bien entendu, des aspects ethnométhodogiques, de par mon immersion dans le groupe étudié. Mon intérêt pour le virtuel et le logiciel libre, ma fréquentation quotidienne de sites, forums et listes de discussion, largement antérieure à la rédaction du présent document, font de moi un observateur privilégié, mais également un acteur à part entière, un membre *lambda* de ces communautés.

L'autre aspect important de ma démarche sera le recours à l'analyse documentaire : un des grands avantages d'un terrain virtuel par rapport à un terrain tangible est l'**accent mis sur l'écrit**. Il est dès lors relativement facile de retrouver des traces documentaires très précises se rapportant à chacun des événements que j'analyserai.

---

1 Ce terme est la traduction directe de l'expression « *In Real Life* », (souvent abrégé par son acronyme « *IRL* ») pour désigner la vie réelle, la vie tangible, par opposition à la vie virtuelle.

Verba volant, scripta manent. La volonté affichée d'ouverture – plus que de volonté, on peut parler de valeur fondamentale – que manifestent les acteurs appartenant à ce mouvement renforce considérablement cet avantage du virtuel sur le tangible : le code source des applications n'est pas le seul à être accessible librement ; les discussions ayant permis de créer, de mettre au point, de corriger ces applications sont également accessibles, archivées, disponibles. Ce contexte social immédiatement accessible nous sert, sur un plateau, un terrain d'une grande richesse – parfois peut-être *trop* riche, la tentation de vouloir tout analyser m'ayant à de nombreuses reprises égaré dans des terrains voisins, mais qui auraient en eux-mêmes pu faire l'objet de plusieurs travaux de l'envergure de ce texte.

Les cas que j'étudierai dans mon analyse empirique (troisième partie de ce document) sont autant d'affaires, disputes, controverses. Ils présentent donc un intérêt sociologique vif et permettent d'être étudiés avec les outils de la sociologie de la critique (Boltanski). Les explications spontanées des registres des acteurs (Callon, Boltanksi) que permet cette méthode les instituent en point d'entrée privilégié pour l'investigation scientifique (Latour).

III. La contrainte linguistique

Ce détail a son importance : l'étude du phénomène nécessite une connaissance, parfois pointue, de la langue anglaise, de son argot, ainsi que d'une sous-catégorie de la langue, le jargon informatique.

Cela ne signifie pas que le phénomène du logiciel libre soit une particularité propre aux pays de culture anglo-saxonne. Au contraire, l'aspect virtuel de ces communautés rend les distances et les frontières relativement anecdotiques. C'est donc assez logiquement que ces communautés ont adopté la « langue d'Internet », l'anglais. Ce choix s'explique par le rôle de plus en plus marqué de l'anglais comme langue internationale (qui permet par exemple à un Finlandais et un Portugais de se comprendre), mais aussi pour des raisons historiques, telles que les origines nord-américaines du réseau ou les difficultés qu'avaient les premiers ordinateurs à afficher les glyphes nécessaires à l'écriture de nombreuses langues autres que l'anglais[2].

Au long de ce document, je proposerai, pour la plupart des citations anglophones, une traduction soit littérale, soit paraphrasée des extraits analysés et j'essayerai, dans la mesure du

2 http://en.wikipedia.org/wiki/Internet#Language

possible, de garder intact l'esprit des textes traduits. Toutes les traductions employées dans ces pages – y compris celles de certains documents dont une traduction française existe (je pense à l'ouvrage d'Himanen, entre autres) – sont de mon cru. Ça et là, quelques précisions linguistiques seront apportées, lorsque la compréhension d'un terme sera cruciale dans le développement d'une idée.

Certains textes en anglais, principalement dans le chapitre consacré à mon étude empirique d'une mailing-list de développeurs, comportent un nombre important de fautes d'orthographe ou de grammaire (beaucoup de contributeurs à cette mailing-list, dont les deux développeurs originaux, n'ont pas l'anglais pour langue maternelle). Plutôt que d'alourdir les extraits d'un nombre considérable de « (sic) », j'ai fait le choix de laisser le texte tel quel. Ces passages étant des « copier/coller », le risque d'introduction d'erreurs de mon propre chef est faible. Les « (sic) » sont donc implicites dans ces pages. Les traductions françaises, bien entendu, n'essaient pas de reproduire ces erreurs.

IV. La problématique en quelques questions

Ce document sera l'occasion de préciser un certain nombre de notions et de répondre à de nombreuses questions concernant le phénomène du logiciel libre. Ces questions sont nettement moins triviales qu'elles ne le paraissent au première abord et possèdent une forte pertinence sociale dans le débat actuel relatif au logiciel libre et « *Open Source* ».

Par souci de clarté, j'aurai recours à une énumération assez brute, dénuée de tout artifice littéraire, pour formuler les questions qui seront abordées dans le présent document.

- Que sont ces communautés ? Qui en sont les membres ?
- Y a-t-il une seule communauté libre, ou un ensemble fragmenté de communautés libres ?
- Y a-t-il des valeurs partagées unanimement ? Une morale ?
- Au sein de ces communautés, y a-t-il des valeurs qui séparent, des intérêts divergents ?
- Ces communautés remettent-elles en cause le capitalisme ? De l'autre côté de l'Atlantique, principalement, des voix s'élèvent pour mettre le public en garde contre le mouvement libre, rapproché du communisme. Il y est parfois qualifié d'« anti-américain ». Qu'en est-il ?

- Quels sont les intérêts poursuivis par les différentes factions de ces communautés ?
- Ces communautés se définissent-elles « contre » quelque chose, « pour » quelque chose ? Les deux ?
- Qui se trouve à la tête de ces communautés ?
- Quelles sont les qualités requises pour devenir membre de ces communautés ?
- Comment s'organisent les membres de ces communautés ?
- Quels sont les rapports entre les têtes pensantes et les citoyens de ces communautés ?
- Quels sont les outils au service de ces valeurs ?
- Comment vivent-elles leur caractère virtuel ?
- Comment se manifeste l'attachement de ces communautés à des valeurs communes ?
- Quand ces communautés sont-elles apparues ?
- Sommes-nous confrontés à un phénomène de mode ?

Partie I
Le logiciel libre

I. Quelques considérations préliminaires

Le phénomène que nous allons étudier est celui du logiciel libre, désigné parfois comme « *free software* » (logiciel libre), parfois comme logiciel « *Open Source* » (dont la source est ouverte). Nous verrons plus loin l'origine de ces deux termes.

Dès à présent, il faudra prendre en compte la **polysémie du terme anglais *free***, qui signifie non seulement « **gratuit** », mais surtout, dans le cas du *free software*, « **libre** ». Cette polysémie est la cause de nombreuses confusions dans le chef de nombreux utilisateurs, qui auront tendance à considérer la gratuité comme la caractéristique première du *free software*.

Open source et *free software* ne sont pas exactement synonymes l'un de l'autre, principalement pour des raisons de luttes fratricides entre les partisans du premier, qui accusent souvent les seconds d'être trop radicaux.

Il est toutefois à noter que ces disputes sont surtout le fait des têtes pensantes du mouvement – les *gourous*, comme ces derniers sont habituellement désignés – et que la plupart des membres de ces communautés emploient ces deux termes indifféremment, de manière interchangeable. Un terme générique a été créé dans le but de désigner l'ensemble de ces logiciels tout en transcendant ces querelles d'école : l'acronyme « *FLOSS*[3] », pour « *Free/Libre/Open Source Software* ». On peut remarquer que le terme anglophone emploie le mot français (ou espagnol) « libre » de manière à désambiguer le terme *free*.

Ces précisions apportées, il me reste à présenter quelques éclaircissements sur la nature du **code source**, libre et ouvert, qui fait la particularité de ce *FLOSS* et que l'on oppose au code source **fermé**, **propriétaire**. Pour comprendre l'enjeu de la disponibilité de ce code source, employons une métaphore culinaire : la recette du Coca-Cola est célèbre pour être ja-

3 On retrouve parfois le terme écrit « *F/OSS* » ou encore tout simplement « *FOSS* », parce que l'acronyme « *FLOSS* » est malheureusement synonyme avec le mot « *floss* », qui signifie « fil dentaire ».

lousement gardée secrète par la Coca-Cola Company et la possession d'une bouteille de Coca-Cola, le produit fini, ne permet pas d'en déduire la recette.

Sans entrer dans trop de technicité et afin de s'entendre sur une définition opératoire du code source, on peut dire que celui-ci correspond, à une recette, qui permet de générer le produit fini, le « code binaire » (ou l'« exécutable »). Ce dernier est, pour continuer notre métaphore culinaire, l'équivalent du plat servi à table : le programme tel que le comprendra l'ordinateur, mais qui est très difficilement déchiffrable par un être humain (cette difficulté tend vers l'impossibilité au fur et à mesure que le programme acquiert une certaine ampleur).

Les partisans du code source libre ou ouvert postulent que le code source doit être fourni avec le code exécutable – ou du moins qu'il doit être possible de se procurer ce dernier afin de l'étudier, le modifier ou le corriger. Cela va à l'encontre du modèle « closed source », ou propriétaire, prédominant dans l'industrie du logiciel au cours des années 1980 et 1990, et défendu par des éditeurs tels que Microsoft, Adobe ou Macromedia. L'archétype de ce modèle est le très célèbre Windows, édité par Microsoft, qui en conserve le code source aussi jalousement que Coca-Cola conserve sa recette.

Notons que tant Microsoft qu'Adobe ou Macromedia fournissent chacun différents logiciels tout à fait gratuits (citons *Internet Explorer* et *Microsoft Media Player* pour Microsoft, *Adobe Acrobat Reader* pour Adobe ou *Flash Player* pour Macromedia). Comme je l'expliquais plus haut, ces logiciels ne sont *free* que dans l'acception « gratuit » du terme et ne seront jamais considérés comme *free software*, la composante la plus importante du *free*, la liberté du code source, leur faisant défaut.

Enfin, je préciserai que le modèle du logiciel libre a déteint sur des domaines qui dépassent celui du logiciel. Ainsi, la démonstration la plus éclatante de l'applicabilité du modèle à d'autres domaines est la Wikipédia, une encyclopédie entièrement libre, réalisée selon un modèle calqué sur celui du logiciel libre.

II. Présentation des licences

Ces principes posés, encore fallait-il les formaliser, les couler dans un texte qui en refléterait l'esprit et en imposerait le respect[4]. Cette lourde tâche revient aux licences logicielles,

4 « *Tentant de régler les usages dans la cité du logiciel, les* hackers *valorisent un type particulier de*

généralement représentées par un petit fichier texte, distribué avec le programme qui y est soumis.

Les licences *FLOSS* ne manquent pas. Si certaines d'entre elles se contentent d'énoncer quelques principes, de donner quelques conseils, quelques lignes de conduites, assorties de quelques babioles juridiques, d'autres prennent l'allure de véritables **manifestes du libre**. La lecture d'une licence permet presque toujours de situer la position de ses auteurs quant à la définition du libre, de l'ouvert, quant aux libertés qu'ils souhaitent donner aux utilisateurs du logiciel qu'elles accompagnent.

Dans les paragraphes suivants, nous allons analyser la GPL (licence type du *copyleft fort*) et la licence BSD (licence dépourvue de *copyleft*)

1. Copyleft : la GPL

Parmi ces licences, l'une des plus célèbres et des plus médiatisées est la *GNU General Public License*, version 2, datée de juin 1991. Son texte compte parmi les plus commentés, mais aussi les plus controversés des écrits relatifs au logiciel libre. Son auteur, *Richard M. Stallman* (*RMS*), fondateur de la *Free Software Foundation* (*FSF*), est l'un des gourous les plus en vue dans le monde du libre (voir ci-dessous). L'une des raisons du succès de cette licence est que le logiciel libre le plus connu, le système d'exploitation Linux, dont la première version publique remonte au 5 octobre 1991, est distribué selon les termes de cette licence.

Le texte de la GPL se compose d'un préambule, de treize articles et d'un exemple d'utilisation. Si ses articles décrivent très précisément les droits et obligations des utilisateurs des logiciels qui y sont soumis, son préambule nous informe d'une manière plus littéraire des intentions de son auteur :

> « The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users (...) »

---

régulation caractérisé par la prééminence des inscriptions morales. » (Auray, *De l'éthique à la politique : l'institution d'une cité libre*, 2002)

(« Les licences de la plupart des logiciels sont conçues pour vous retirer la liberté de les partager et de les modifier. Par contraste, la Licence Publique Générale GNU est destinée à garantir votre liberté de partager et de modifier les logiciels libres -- de s'assurer que le logiciel est libre pour tous ses utilisateurs (...) »)

Les termes employés dans ce texte sont forts et ne s'embarrassent pas de formalités, ne cherchent à ménager personne : les licences de la plupart des logiciels – entendez des logiciels commerciaux – sont, par leur conception même, mauvaises, malignes, maléfiques (« *evil* » est le terme généralement employé par Richard Stallman et ses suiveurs pour désigner ces licences, ou la société qui en représente la pire illustration, selon lui, à savoir Microsoft). Le but de ces licences est de **retirer à l'utilisateur des droits considérés comme élémentaires** : **partager** et **modifier** les logiciels qu'elles accompagnent.

Le second paragraphe de ce préambule, quant à lui, est probablement celui qui est le plus méconnu – ou le plus mal compris :

« When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things. »

(« Quand nous parlons de logiciel libre, nous faisons référence à la liberté, pas au prix. Nos Licences Générales Publiques sont conçues pour vous assurer la liberté de distribuer des copies du logiciel libre (et de faire payer ce service si vous le désirez), [vous assurer] que vous en recevrez le code source, ou pourrez l'obtenir si vous le voulez, que vous pourrez modifier le logiciel ou en utiliser des parties dans vos nouveaux programmes libres ; et que vous sachiez que vous pouvez faire ces choses. »)

Nous retrouvons ici une explication de la manière dont la GPL entend le mot *free* : la liberté, non le prix. Contrairement à une croyance répandue, **la GPL n'implique pas la gra-**

tuité : tout au plus cette gratuité est-elle un effet secondaire heureux de la liberté que permet – qu'impose – la licence.

Nous reviendrons au paragraphe suivant sur les modalités d'application qu'implémente cette licence, mais remarquons d'abord l'avant-dernier paragraphe de ce préambule :

> « Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all. »
>
> (« Finalement, tout programme libre est constamment menacé par les brevets logiciels. Nous souhaitons éviter le danger que les distributeurs d'un programme libre obtiennent des brevets logiciels sur celui-ci, rendant par ce fait le programme propriétaire. Pour éviter cela, nous avons clairement stipulé que tout brevet doit être licencié pour l'utilisation libre de chacun ou [que le logiciel ne doit pas être breveté] du tout. »)

Ce paragraphe attire l'attention sur un problème qui tient particulièrement à coeur aux partisans du logiciel libre : le refus strict de toute brevetabilité logicielle. Cette thématique, qui revient de manière récurrente dans l'actualité, est un sujet particulièrement chaud à l'heure où j'écris ces lignes, puisque, début juillet, le Parlement européen a finalement rejeté une directive sur les brevets logiciels, qui visait à doter l'Europe de ces brevets logiciels semblables à ceux qui existent aux États-Unis d'Amérique[5]. Cette directive – considérée comme une épée de Damoclès pendue au-dessus de la tête des développeurs libres – alimentait les discussions entre partisans du logiciel libre depuis plusieurs mois. Ces discussions avaient pris une telle ampleur que les pages économiques de nombreux journaux de la presse non-spécialisée y

5 « Le rejet de la directive sur la brevetabilité des inventions assistées par ordinateur a été saluée mercredi aussi bien par l'industrie informatique que par les partisans des logiciels libres. (...)
Pour les partisans des logiciels libres, le vote représente au contraire *«une grande victoire de ceux qui se sont battus pour veiller à ce que l'innovation et la compétitivité soit protégée des brevets sur les logiciels »*.
«Elle marque la fin d'une tentative de la Commission européenne de légitimer les pratiques à l'américaine de l'Office européen des brevets », a poursuivi une association FFII, qui milite pour la libre circulation des idées. » (*La Libre Belgique*, 6 juillet 2005)

avaient fait écho, ce qui, en soi, avait déjà été considéré comme une victoire du camp des « libristes ».

Les autres articles de ce préambule implémentent le caractère « viral » de la GPL : ce terme, inventé par l'un de ses adversaires les plus farouches, Microsoft, est expliqué sur le site Internet du géant américain du logiciel :

> « *Restrictive free-software licenses, by contrast, also make source code freely available, but are designed specifically to prevent commercial software development. They accomplish this objective by requiring that all derivative works be licensed under the same terms as the original source code. These licenses often are described as "viral," because they purport to extend their own terms to, or to "infect," software other than the licensed program, and even to software not yet in existence. The most common examples of this type of license are the GNU General Public License (GPL).*[6] »
>
> (« *Les licences de logiciel libre restrictives, par contraste, rendent aussi le code source librement disponible, mais sont conçues spécifiquement pour empêcher le développement de logiciels commerciaux. Ils accomplissent cet objectif en imposant que tous les travaux dérivés soient licenciés sous les mêmes termes que le code source original. Ces licences sont souvent décrites comme « virales », parce qu'elles prétendent étendre leurs propres termes, ou « infecter », les logiciels autres que le programme licencié, et même à des logiciels qui n'existent pas encore. Les exemples les plus communs de ce type de licence sont la Licence Générale Publique (GPL).* »)

Il est à noter que, si le caractère « viral » de la GPL est bien expliqué dans cet extrait, Microsoft confond – ou plus vraisemblablement fait mine de confondre – « logiciels commerciaux » et « logiciels propriétaires » : comme nous venons de le voir dans l'analyse du deuxième paragraphe du préambule, la liberté n'implique pas la gratuité et de nombreux logiciels commerciaux sont basés sur des licences libres, comme les offres Linux de Red Hat, ou même du géant américain IBM.

Les deux derniers paragraphes (numérotés 11 et 12), rédigés en majuscules d'imprimerie, informent l'utilisateur que le programme est fourni « tel quel » et qu'aucune garantie

6 http://www.microsoft.com/resources/sharedsource/Articles/LicensingOverview.mspx

n'est assurée quant à la qualité et aux effets du programme. Ils déchargent également l'auteur du logiciel de toute responsabilité en cas de dommage causé par l'utilisation du programme.

On peut voir dans la GPL une forme de contrainte, une gardienne « virale » de la morale du projet GNU. Pour reprendre l'analyse de Bruno Latour[7], qui voit la ceinture de sécurité comme un moyen d'inscrire une règle morale dans la voiture (délégation morale à un objet), la GPL devient une délégation morale à un texte de type juridique, qui permet à l'auteur d'un code informatique d'être assuré que sa création restera libre pour l'éternité – ou, plus modestement, pour la durée de vie dudit programme et de ses dérivés éventuels.

De même que la ceinture de sécurité de Latour avait comme programme « *ne passez pas à travers le pare-brise* », le programme « *tu garderas ton code libre et celui qui voudra (ré)utiliser et modifier ce code devra à son tour maintenir la liberté de son code* » est imposé par la GPL.

La licence GPL fait partie des licences dites *copyleft* (littéralement « copie gauche » ou « copie laissée » : elles utilisent d'une façon créative les lois sur le *copyright* (littéralement « droit de copie » ou « copie droite »), qui sont normalement utilisées pour restreindre les droits de copie ou de création de versions dérivées, pour s'assurer que tous auront le droit de copier ou de créer des versions dérivées de l'oeuvre. Le *copyleft* peut donc être vu comme le contraire du copyright[8].

2. Non-copyleft : la licence BSD

La licence BSD[9] représente l'autre licence majeure du logiciel libre. Contrairement à la GPL, presque toujours utilisée telle quelle, la licence BSD a été déclinée dans un très grand nombre de variantes, aux effets souvent semblables. On parlera donc plus précisément de licences de type BSD. Toutefois, les différences entre ces licences étant très faibles, voire anecdotiques, et mon but n'étant pas d'entrer dans ce genre de détails techniques, je les désignerai dans ce texte comme « la licence BSD ».

---

7 Latour, *Petites leçons de sociologie des sciences*, 1993, pp. 25 et suivantes et pp. 75 et suivantes.

8 http://en.wikipedia.org/wiki/Copyleft

9 *BSD* est l'acronyme de *Berkeley Software Distribution*, un système dérivé d'UNIX, créé dans les années 1970 et dont certaines déclinaisons sont encore largement utilisées de nos jours (FreeBSD, OpenBSD, NetBSD). À la fin des années 80, une version de BSD fut rendue disponible librement sous les termes de l'ancêtre de ce que nous connaissons aujourd'hui comme la licence BSD.

La licence BSD n'est pas une licence *Copyleft*. Certains la désignent, surenchérissant sur le jeu de mot qui créa le mot *Copyleft*, comme une licence *Copycenter*, à la suite de Kirk McKusick, un programmeur vétéran, qui a contribué à divers systèmes d'exploitation BSD et contribue toujours aujourd'hui au développement de son descendant, FreeBSD. Ce dernier avait décrit la licence BSD en 1999 en ces termes :

> « *The way it was characterized politically, you had copyright, which is what the big companies use to lock everything up; you had copyleft, which is free software's way of making sure they can't lock it up; and then Berkeley had what we called 'copycenter', which is 'take it down to the copy center and make as many copies as you want'.* »
> (« *de la manière dont ce fut caractérisé politiquement, vous aviez le copyright, qui est ce que les grandes compagnies utilisent pour tout fermer ; vous aviez le copyleft, qui est la manière qu'a le logiciel libre de s'assurer qu'elles ne peuvent pas les fermer ; et puis, vous aviez Berkeley qui avait ce que nous avons appelé « copycenter », qui signifie « amenez-les au centre de photocopie et faites-en toutes les copies que vous voulez ».* »).

Les termes de la licence BSD sont assez simples et permissifs : la version actuelle de cette licence, qui ne comporte que trois articles, dit simplement que la notice de copyright qui accompagne le logiciel doit rester intacte et que le nom du détenteur du copyright original (l'université de Berkeley et ses membres) ne peut être utilisé à titre publicitaire. Suit une décharge légale en majuscule d'imprimerie, semblable dans sa formulation et ses effets à celle de la GPL.

On pourrait donc résumer par cette courte formule le programme de cette licence : « *tu feras ce que tu veux de ce code, tant que tu en garderas la notice de copyright intacte* ».

Nous voyons que la licence BSD est particulièrement permissive, même si elle impose un minimum de formes à respecter, principalement dans le but qu'une tierce partie ne s'attribue pas la parenté du code visé par ce texte. Certains auteurs préfèrent renoncer à tout copyright sur leur création et se contentent de « lâcher leur code dans la nature ».

Samuel Hocevar, un auteur de logiciels français, a été jusqu'à publier une licence en réaction à la bataille entre la GPL et la licence BSD, considérées comme maladroites. Si le

contenu de la « Do What The Fuck You Want To Public License » est clairement parodique et sa formulation quelque peu grivoise, ses motivations ne sont pas dénuées d'intérêt :

> « *There is a long ongoing battle between GPL zealots and BSD fanatics, about which license type is the most free of the two. In fact, both license types have unacceptable obnoxious clauses (such as reproducing a huge disclaimer that is written in all caps) that severely restrain our freedoms. The WTFPL can solve this problem.*
> *When analysing whether a license is free or not, you usually check that it allows free usage, modification and redistribution. Then you check that the additional restrictions do not impair fundamental freedoms. The WTFPL renders this task trivial: it allows everything and has no additional restrictions. How could life be easier? You just DO WHAT THE FUCK YOU WANT TO*[10]. »
> (« *Il existe une longue et incessante bataille entre les zélotes de la GPL et les fanatiques de [la licence] BSD, qui cherchent à savoir quelle licence est la plus libre des deux. En fait, les deux types de licences ont des clauses inacceptables et maladroites (comme le fait de faire reproduire une immense décharge légale écrite toute en majuscules) qui restreignent sévèrement nos libertés. La WTFPL peut résoudre ce problème.*
> *Quand vous analysez si une licence est libre ou non, vous vérifiez habituellement qu'elle autorise les libres utilisation, modification et redistribution. Puis vous vérifiez que les restrictions additionnelles n'altèrent pas les libertés fondamentales. La WTFPL rend cette tâche insignifiante : elle autorise tout et n'a aucune restriction additionnelle. Comment la vie pourrait-elle être plus facile ? VOUS FAITES SIMPLEMENT CE QUE VOUS VOULEZ FAIRE.* »)

Cette licence, qui adopte la forme habituelle des licences informatiques (un fichier texte annexé au programme, un titre, un numéro de version, un copyright (!) et une autorisation de dupliquer la licence elle-même, tant que celle-ci n'est pas changée), ne dit rien de plus que son titre : « `0. You just DO WHAT THE FUCK YOU WANT TO.` »

10 http://sam.zoy.org/wtfpl/

3. Comparaison entre les licences libres

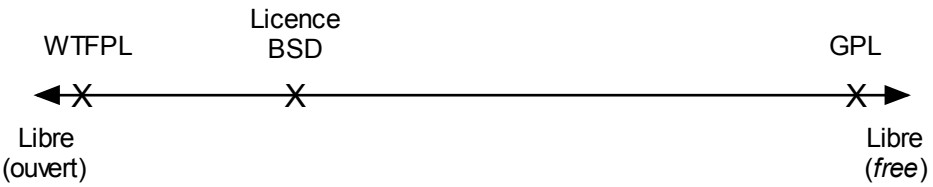


Fig. 1 : les deux pôles du libre

Ce dernier exemple, certes plus léger, en plus de nous permettre de souffler quelque peu après ce très conventionnel exposé des tenants et aboutissants des licences libres, me permet de préciser les deux **valeurs concurrentes** qu'implique la liberté du logiciel.

Tout à droite de l'axe représenté dans la Fig. 1, nous avons la position des défenseurs de la GPL, Richard Stallman et la *Free Software Foundation* en tête, pour qui la liberté d'écrire un logiciel passe par la **certitude qu'à l'avenir, son code source conservera sa liberté**. À l'inverse, tout à gauche de cet axe, nous avons ceux pour qui la liberté passe par un **refus de tout copyright, de tout droit**, par une certaine idée « anarchique » de la distribution de la connaissance. À peu près au milieu de cet axe, mais probablement plus près de sa gauche que de sa droite (les conséquences d'une simple ligne de copyright pouvant être considérées comme beaucoup moins importantes que l'obligation de publier tout code dérivé), nous avons la licence BSD. Déterminer laquelle de ces deux licences est la plus libre – et donc si l'axe de la Fig. 1 se lit de droite à gauche ou de gauche à droite – implique donc de **savoir si la liberté est une valeur intuitive, anarchique et pouvant se passer de formes** (gauche) **ou, au contraire, une valeur qui implique la mise en oeuvre de nombreuses précautions visant à garantir sa pérennité**.

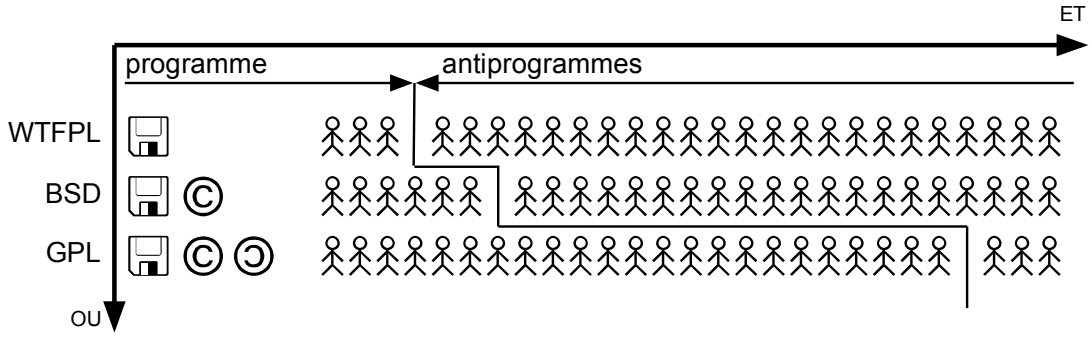


Fig. 2 : programme et antiprogrammes du logiciel libre

Si nous admettons que le programme du logiciel libre est tel que le conçoit Richard Stallman et sa *Free Software Foundation*, à savoir garder un programme et tous ses dérivés libres, nous pouvons tracer le schéma représenté en Fig. 2.

- Une licence très permissive (la parodique WTFPL dans le schéma, mais plus généralement l'absence de licence ou de copyright) donnera très peu de motivation aux programmeurs (ré)utilisant les sources du logiciel à distribuer son programme dérivé de façon libre. **Seul le sens moral de ces programmeurs sera mis en oeuvre pour les inciter à adopter ce comportement**. Ceci implique deux choses : 1. que le créateur de logiciels **souscrive à la morale du logiciel libre** et 2. qu'il **applique cette morale à ses créations**.

- Une licence un peu plus restrictive (BSD) obligera le programmeur à créditer les auteurs du code original (ou, vu à l'envers, lui interdira de s'attribuer les mérites de la création originale). Cela **ne l'obligera en rien à « jouer le jeu »** du logiciel libre, mais **pourra donner un coup de pouce à son sens moral**. C'est ainsi que fonctionnent de nombreux projets basés sur une licence de type BSD (FreeBSD, Apache, l'environnement X Window, PHP).

- La licence GPL, au contraire, **garantira la morale du logiciel libre** en obligeant le programmeur à y adhérer. Certains ignoreront quand même la licence et emploieront le code dans un logiciel fermé, mais ils seront dans ce cas en **infraction** non seulement à la GPL, mais aussi **aux lois sur le copyright**. À adopter un tel comportement, ils risqueront donc une action en justice. Cette perspective découragera la plupart des développeurs, qui soit adhéreront à la GPL, soit n'utiliseront pas de code GPL dans leurs programmes.

III. Quelques définitions

Bien que mon but soit ici de produire un texte ne demandant aucune connaissance particulière en informatique, j'aurai recours au fil de ces lignes à quelques termes appartenant au langage vernaculaire de l'informatique ou de ma communauté étudiée. En voici quelques définitions.

1. Slashdot

Acteur essentiel du logiciel libre et *Open Source*, Slashdot.org est le site web de technologie le plus fréquenté par les membres de la communauté. Sa page principale, mise à jour à plusieurs reprises par heure, est constituée de liens vers des articles issus de diverses publications en ligne, le plus souvent de nature technologique. Ce lien est toujours associé à un court résumé du contenu de l'article lié, la plupart du temps accompagné d'une brève opinion sur celui-ci.

Le slogan du site est « *News for nerds. Stuff that matters.* » (« *Nouvelles pour les nerds*[11]. *Choses qui comptent.* »). Ces « choses qui comptent » touchent généralement au monde informatique, avec un fort biais positif envers le logiciel *FLOSS* et un biais souvent négatif envers le logiciel propriétaire, avec certaines exceptions qui reflètent la ligne éditoriale du site (par exemple, le traitement des nouvelles relatives à la société informatique Apple est la plupart du temps élogieux).

Le nombre de visiteurs de Slashdot est tel que, lorsqu'un article est lié sur ce site, il n'est pas rare que le serveur hébergeant ledit article sature, jusqu'à devenir indisponible pendant plusieurs heures. Ce phénomène, habituellement désigné comme « l'effet Slashdot », s'explique par l'incapacité de nombreux serveurs web à répondre à des milliers de requêtes par minutes.

Une des raisons du succès de Slashdot est son système de commentaires. Chaque article, en effet, propose un système de commentaires disposant d'un système de modération évolué, permettant une revue des pairs quasi-instantanée : les membres réguliers du site se voient périodiquement attribuer cinq points de modération, qu'ils peuvent utiliser pour noter les contributions au forum (« *posts* »), sur une échelle allant de -1 à +5. Le nombre de *posts* se compte par centaines pour chaque articles, avec des pointes dépassant les 1 000 *posts* lorsqu'un sujet d'actualité particulièrement chaud est traité.

Une lecture critique de Slashdot – et surtout de ses commentaires –, même si elle peut être extrêmement onéreuse en temps, permet de se faire une opinion sur la plupart des sujets d'actualité informatique. Bien souvent, l'article lié n'est que le prétexte à l'ouverture de la dis-

11 « *Nerd* » est un terme souvent employé ironiquement pour désigner un passionné de technologie, faiblement socialisé et ne vivant que pour sa passion.

cussion sur les forums. Il est connu qu'un grand nombre de « *posters* » se rendent directement sur les commentaires de l'article sans avoir lu celui-ci.

Le site est parfois critiqué comme étant un organe de propagande du mouvement *FLOSS*. Il serait effectivement difficile d'étudier le phénomène sans avoir lu les articles – et commentaires – de Slashdot relatifs aux sujets analysés dans cette étude, tant les discussions que l'on peut y lire font autorité dans une frange importante de la communauté libre.

2. Hacker

Ce terme est souvent galvaudé par les médias, qui l'emploient généralement pour désigner toute personne utilisant de près ou de loin l'informatique à des fins illicites – « piratage » informatique, pénétration dans des systèmes bancaires, vol de données confidentielles. Dans le langage vernaculaire de l'informatique, toutefois, ce terme désigne simplement un passionné d'informatique, curieux du fonctionnement de ses ordinateurs et aux connaissances généralement assez larges dans le domaine. C'est dans ce sens qu'il faudra le lire dans le présent document.

3. Mailing-list

Une mailing-list est une liste de distribution de courrier électronique. Il s'agit d'une des méthodes les plus utilisées par les membres d'une communauté quelconque (informatique ou autre) pour interagir entre eux. Le principe est simple : un membre envoie un e-mail à une adresse prévue à cet effet et l'ordinateur (« serveur ») qui reçoit cet e-mail se charge de le rediriger vers les boîtes de courrier électronique de chacun des inscrits à la liste. Les réponses à cet e-mail seront redirigées de même manière à l'ensemble des membres de la liste, créant des « fils de discussion ».

IV. Brève introduction historique

Le logiciel informatique est né dans les années 1960, dans les laboratoires de recherche universitaire et dans quelques entreprises spécialisées. Il naquit libre : il n'était qu'un ensemble d'instructions relativement simples, nécessaires pour utiliser le matériel qu'il accom-

pagnait et les chercheurs qui l'écrivaient formaient une communauté qui s'échangeait librement son code source. Cette tradition s'exprimait sous la forme d'un droit coutumier[12].

Au cours des années 1970, le logiciel informatique gagna peu à peu en complexité. Parti d'un simple jeu d'instructions (logiciel de bas niveau, que l'on désignera par après par le terme « système d'exploitation »), il gagna en complexité. Des applications dédiées à certaines tâches apparurent (bases de données, traitements de texte, tableurs, etc.).

Vers le début des années 80, les distributeurs de ces logiciels en arrivèrent à ne plus fournir que la forme compilée du programme, gardant son code source comme un secret professionnel.

V. Gourous

1. Richard Stallman

1. Idéologie

> « *This meant that the first step in using a computer was to promise not to help your neighbor. A cooperating community was forbidden. The rule made by the owners of proprietary software was, "If you share with your neighbor, you are a pirate. If you want any changes, beg us to make them.*"[13] »
> (« *Cela signifiait que le premier pas menant à l'utilisation d'un ordinateur était de promettre de ne pas aider son prochain. Une communauté qui coopérait était interdite. La règle établie par les propriétaires de logiciels fermés était « Si tu veux partager avec ton prochain, tu es un pirate. Si tu veux un quelconque changement, supplie-nous de le faire.* » »).

C'est en ces termes que *Richard Stallman* décrit le phénomène de « propriétarisation » qui se généralisa dans le monde du logiciel au début des années 80. Stallman, né le 15 mars 1953, diplômé en physique de l'université de Harvard en 1974, commença à fréquenter le la-

12 « *La volonté de circulation libre des routines dans le milieu des informaticiens est issue d'une tradition qui s'exprimait auparavant non pas sous la forme juridique, mais sous la forme de droits coutumiers, bafoués par l'apparition des systèmes de propriété intellectuelle (brevets, droits d'auteur, secrets commerciaux) dans les années quatre-vingt.* » (Auray, *Le savoir en réseaux et l'empreinte inventive*, 2000)

13 Stallman, *About the GNU project*, 1998

boratoire d'intelligence artificielle de l'Institut de technologie du Massachusetts (MIT) en 1971.

> « *The idea that the proprietary-software social system--the system that says you are not allowed to share or change software--is antisocial, that it is unethical, that it is simply wrong, may come as a surprise to some readers. But what else could we say about a system based on dividing the public and keeping users helpless? Readers who find the idea surprising may have taken proprietary-software social system as given, or judged it on the terms suggested by proprietary software businesses. Software publishers have worked long and hard to convince people that there is only one way to look at the issue.*
> *When software publishers talk about "enforcing" their "rights" or "stopping piracy", what they actually *say* is secondary. The real message of these statements is in the unstated assumptions they take for granted; the public is supposed to accept them uncritically. So let's examine them.*[14] »
> (« *L'idée que le système social du logiciel propriétaire – le système qui dit que vous n'êtes pas autorisés à échanger ou modifier le logiciel – est antisocial, anti-éthique et tout simplement injuste, pourrait surprendre certains lecteurs. Mais que pourrait-on dire d'autre sur un système basé sur la division du public et qui laisse les utilisateurs livrés à eux-mêmes ? Les lecteurs qui jugent l'idée surprenante considèrent peut-être le système social du logiciel propriétaire comme un donné, ou bien le jugent sur base des termes suggérés par les entreprises qui développent du logiciel propriétaire. Les éditeurs de logiciels ont oeuvré longuement et durement à convaincre le monde qu'il n'y a qu'une manière de considérer le problème.*
> *Quand les éditeurs de logiciels parlent de « faire respecter » leurs « droits » ou de « stopper le piratage », ce qu'ils *disent* réellement est secondaire. Le vrai message porté par ces déclarations se trouve dans les assomptions non-dites qu'ils considèrent acquises. Le public est supposé les accepter de manière non critique. Nous allons donc les examiner.* »).

Ces trois assomptions, telles que décrites par Stallman comme profondément fallacieuses et injustes, sont :

14 Stallman, *About the GNU project*, 1998

- qu'il existe un **droit naturel** – et donc impossible à remettre en question – de **posséder un logiciel** et donc d'**avoir du pouvoir sur ses utilisateurs** ;
- que la seule chose importante est que ce système crée de l'emploi et que les utilisateurs d'ordinateurs **ne doivent pas s'inquiéter du modèle de société que ce système crée** ;
- que nous n'aurions pas de logiciels efficaces si on n'attribuait pas ce **pouvoir des entreprises sur les utilisateurs** du programme.

Il rejette ces assomptions en analysant les types d'arguments employés par les vendeurs de logiciels pour asseoir leur pouvoir de contrôle sur la manière dont est utilisée l'information. Il classe ceux-ci en cinq catégories[15] :

- l'emploi de **termes inappropriés**, comme par exemple le mot « vol », employé pour guider l'utilisateur à penser selon une « analogie simpliste » entre des programmes et des objets physiques ;
- le recours à des **exagérations** pour décrire les pertes occasionnées par la duplication non-autorisée de logiciels ;
- l'**état actuel du droit** : il pense que le droit actuel n'est pas moral et fait le parallèle avec la ségrégation raciale, qui faisait partie du droit américain quarante ans plus tôt ;
- le **recours au droit naturel** : comme nous l'avons vu plus haut, Stallman pense que le système du copyright est contraire au droit naturel et que le recours à celui-ci par les apologistes du logiciel propriétaire est donc fallacieux ;
- l'**argument économique**, auquel il reconnaît une certaine légitimité, mais auquel il trouve une faille : la qualité d'un logiciel n'est pas liée au prix qu'il coûte. Pour Stallman, cet argument découle également de l'analogie entre le logiciel et les biens matériels.

Nous le voyons, les idées, la philosophie de Stallman s'inscrivent dans plusieurs champs : le champ **technique**, bien entendu, mais aussi les champs **économique**, **politique**, **juridique**, **moral et éthique** et même **religieux** (« aider son prochain ») – même s'il se considère lui-même comme athée. Ses positions tranchées lui valent d'être parfois considéré comme un intégriste du logiciel libre et suscitent régulièrement des controverses même au sein des défenseurs de celui-ci, comme nous le verrons un peu plus tard.

15 Stallman, *Why Software Should Not Have Owners*, 1998

2. Naissance du projet *GNU* et de la *Free Software Foundation*

Comme nous l'avons vu dans le court historique exposé quelques lignes plus haut, le tournant des années 80 a marqué une rupture chez les développeurs, qui **passèrent d'un modèle basé sur l'échange à un modèle basé sur le secret et la propriété intellectuelle**. Stallman se retrouva confronté à un choix : abandonner toute activité informatique, adhérer aux valeurs du logiciel propriétaire ou combattre celui-ci en créant un mouvement alternatif. Il choisit cette troisième solution : « *So I looked for a way that a programmer could do something for the good. I asked myself, was there a program or programs that I could write, so as to make a community possible once again?* » (« *Alors je cherchai comment un programmeur pourrait faire quelque chose pour le* ***bien****. Je me suis demandé : existe-t-il un ou des programmes que je pourrais écrire, de manière à rendre de nouveau possible une communauté ?* »).

La réponse lui parut comme une évidence : un système d'exploitation. Non seulement parce que c'était un domaine qu'il connaissait et maîtrisait, mais aussi parce que le système d'exploitation est un ensemble de logiciels de bas niveau, sans lequel il n'est tout simplement pas possible de démarrer un ordinateur. Attaquer la problème à la base était donc logique.

Nous étions en 1983 lorsque Stallman lança le « *Free Software Movement* » (« *Mouvement du Logiciel Libre* »). Il décida de baptiser son projet « *GNU* », acronyme récursif de « *GNU's Not Unix* » (« *GNU n'est pas Unix* »). Le projet, ambitieux, visait à réécrire un système complet calqué sur le système Unix, mais débarrassé de ses contraintes propriétaires. Pour l'aider dans cette tâche, il fonda en 1985 la « *Free Software Foundation* » (« *Fondation du Logiciel Libre* »), une association caritative destinée à promouvoir le logiciel libre et le projet GNU. Stallman, rejoint par différents développeurs, développa plusieurs logiciels pour le projet GNU (*GNU/Emacs*, un éditeur de fichiers à tout faire, *GCC*, un compilateur, *BASH*, un interpréteur de commandes, etc.). Ensuite, il formalisa sa position sur la liberté du logiciel, dans la première version de la GPL, parue en 1989. En 1991, une seconde version de cette licence parut, qui visait à rendre ses termes moins équivoques. C'est de cette version 2 de la GPL que j'ai longuement parlé au chapitre précédent[16].

16 À l'heure où j'écris ces lignes, Stallman et Eben Moglen, un professeur de droit proche de la *Free Software Foundation*, travaillent sur une version 3 de la GPL, qui tente d'adapter la GPL à l'évolution qu'a connu l'informatique depuis 1991. Cette tâche serait particulièrement malaisée et aucune date n'a été arrêtée à ce jour pour la publication de ce texte.

Depuis les années 1990, la principale activité de Stallman est devenue celle de porte-parole du logiciel libre et de la *Free Software Foundation*. Sa désignation comme « gourou » du logiciel libre n'est pas due au hasard : si nous avons vu plus haut que son premier but était d'*aider son prochain*, il entretient lui-même l'image du logiciel libre vu comme une religion – ou une secte. Ainsi, se présente-t-il parfois comme « *Saint IGNUcius, de l'Église d'Emacs* ».

> « *Richard Stallman, the bearded and longhaired guru, attends computer gatherings in a robe, and he exorcises commercial programs from the machines brought to him by his followers.*» (Himanen, 2001, p. 5)
> (« *Richard Stallman, le gourou barbu et aux cheveux longs, assiste aux rassemblements informatiques en robe et exorcise les programmes commerciaux des machines apportées par ses disciples.* »).

Fig. 3 : Richard Stallman, dans la tenue du gourou « Saint-IGNUcius »

> « *To join the Church of Emacs you must repeat the confession of the faith three times. You must say, "There's no system other than the GNU and Linux is one of its kernels." (Laughter)*
> *The Church of Emacs has some advantages compared with some other churches because to be a Saint in the Church of Emacs does not require celibacy. (Laughter)*
> *However it does require making a moral commitment to live a life of purity and then living by it.*
> *You must exorcise the evil proprietary operating systems from all the computers you control. And then install a Wholly free operating system instead. Because Wholly can be spelled in more than one ways.*
> *And then only install free software on top of that.*
> *If you make this commitments and live by it, then you too would be a saint and you*

too may eventually have a halo, if you can find one because we don't make them anymore. (Laughter)[17] »

(« *Pour rejoindre l'Église d'Emacs, tu dois répéter trois fois la confession de foi. Tu dois dire « Il n'y a pas d'autre système que le GNU et Linux est un de ses noyaux ». (Rires)*

L'Église d'Emacs a quelques avantages, comparée avec d'autres églises, parce qu'être un Saint dans l'Église d'Emacs ne demande pas le célibat. (Rires)

Cependant, cela nécessite l'engagement moral de vivre une vie de pureté, et de suivre cet engagement.

Tu dois exorciser les systèmes d'exploitation propriétaires maléfiques de tous les ordinateurs que tu as sous ton contrôle. Puis installer un (entièrement/Saint) système d'exploitation à leur place. Parce que entièrement/Saint[18] *peut s'écrire de plusieurs manière.*

Puis, ensuite, n'installer que du logiciel libre sur ces machines.

Si tu fais ces engagements et les suis, tu seras aussi un Saint et tu pourras finalement avoir un halo, si toutefois tu en trouves un, parce qu'on n'en fabrique plus[19]*. (Rires)* »)

Même si ces propos sont évidemment à prendre au second degré, le choix de termes à connotation mystique n'en est pas moins éclairant quant aux motivations de Stallman et vient compléter notre tableau des valeurs auxquelles souscrit l'idéologie GNU, déjà abordée dans mon analyse de la GPL et dans la première section du présent chapitre.

3. Une contribution sociale

Stallman se sent **investi d'une mission**, la protection des utilisateurs informatiques contre les entreprises de développement de logiciel, coupables d'empêcher le public d'accéder à la connaissance. L'**échange**, le fait d'« aider son prochain », est inscrit dans les textes fondateurs de l'« Église » de Stallman (en l'occurrence la GPL). Le **Saint** est celui qui aura fait ce voeu de répandre le **bien** autour de lui, de souscrire à la **morale** du logiciel libre.

---

17 Transcription de l'intervention du 13 Mars 2002 de Richard Stallman à l'Institut de Technologie de Madras, Chrompet, Chennai (Inde).

18 Les mots « *holly* » et « *wholly* », en anglais, sont homophones et se traduisent respectivement pas « saint » et « entièrement ».

19 Le « halo » arboré par Stallman est en fait l'ancien plateau d'un disque dur, d'un diamètre d'une cinquantaine de centimètres. De tels disques ne sont plus fabriqués depuis quelques décennies (Fig. 3).

Chez Stallman, la **liberté** s'inscrit dans un cadre beaucoup plus large que ses simples effets sur les logiciels, puisque tout comportement visant à protéger le logiciel est un principe fondamentalement **antisocial**, **anti-éthique**. Il peut, de prime abord, sembler étrange d'inscrire cette liberté dans un cadre si large, qui dépasse de loin le monde des logiciels. Ce serait oublier que chez Stallman, la publication de code informatique sous une licence libre est une importante **contribution sociale**. *A contrario*, ne pas adopter ce comportement est un acte nuisible envers l'humanité :

> « *Extracting money from users of a program by restricting their use of it is destructive because the restrictions reduce the amount and the ways that the program can be used. This reduces the amount of wealth that humanity derives from the program. When there is a deliberate choice to restrict, the harmful consequences are deliberate destruction.*[20] »
> (« *Soustraire de l'argent aux utilisateurs d'un programme en restreignant leur capacité à l'utiliser est destructeur, parce que ces restrictions réduisent les manières dont un programme peut être utilisé.* ***Cela réduit la quantité de richesses que l'humanité peut tirer du programme. Quand il y a un choix délibéré de restreindre, les conséquences nuisibles constituent une destruction délibérée.*** »).

2. Linus Torvalds

Personnage clef du logiciel libre, Linus Torvalds, né le 28 décembre 1969, est le père d'un des logiciels libres les plus utilisés, le système d'exploitation Linux. Linux est incontestablement le logiciel libre le plus connu du grand public, qui rattache souvent la notion intuitive qu'ils ont du logiciel libre ou *Open Source* à Linux.

1. Linux, précisions terminologiques

Une petite précision s'impose à ce stade : Linux désigne deux réalités différentes, selon qu'on l'utilise au sens strict ou au sens large.

– *Sensu stricto*, Linux est une couche logicielle de bas niveau, qui se charge du rôle d'intermédiaire entre le *hardware* (matériel informatique, tel que le microprocesseur, le disque dur, la mémoire, etc.) et les logiciels (*software*) de plus haut niveau (système de fenê-

20 Stallman, *The GNU Manifesto*, 1985

trage, systèmes de gestion de bases de données, programmes de traitement de texte, serveurs de fichiers, etc.). On parle alors, du « *kernel* » Linux (« *noyau* »). Un noyau, en tant que tel, n'est pas suffisant pour utiliser un ordinateur.

– *Sensu lato*, Linux désigne une « distribution » de logiciels, c'est-à-dire le noyau lui-même, accompagné de dizaines ou de centaines d'autres programmes de plus haut niveau, qui rendent l'ordinateur utilisable.

Linus Torvalds est responsable du noyau de Linux au sens strict. Linux au sens large utilise principalement les outils GNU pour compléter le noyau fourni. C'est pourquoi Richard Stallman milite pour que ces distributions soient appelées « GNU/Linux » au lieu de Linux, le noyau représentant, quantitativement, une partie très faible de ces distributions.

2. Linux, position dans le projet GNU

En 1990, lorsque Linus Torvalds s'offre un ordinateur personnel de type Intel 80386, il est désabusé par les systèmes d'exploitation alors disponibles sur ces machines. Les offres MS-DOS de Microsoft sont peu attractives, les Unix commerciaux sont inabordables et le noyau du projet GNU, GNU/HURD, ne devrait pas arriver à maturité avant deux bonnes années[21]. Torvalds se mit donc à développer son propre noyau, sans ambition particulière, « *just for fun* » (« *juste pour l'amusement* »).

De fil en aiguille, il en arrive à publier son noyau sur le serveur Internet d'un de ses amis, qui nomme la section de son serveur FTP réservée au noyau de Torvalds « *Linux* », un jeu de mots évident entre « Unix » et le prénom de Torvalds (traditionnellement, les systèmes de type Unix portent un nom dont la dernière lettre est un x).

Pendant les quelques premiers mois de sa vie, les sources de Linux sont distribuées sous un simple copyright, sans licence associée. Très vite cependant, cette restriction devient problématique, lorsque des contributeurs externes souhaitent apporter des modifications à Linux. Torvalds décide alors de le placer sous une licence libre et, par reconnaissance envers

21 Aujourd'hui, en 2005, même si GNU/HURD a fait quelques progrès, il n'est toujours pas arrivé à un niveau de maturité suffisant pour être utilisable. La principale raison est que Linux a attiré à lui la plupart des développeurs potentiels de GNU/HURD, rendu redondant par l'existence même de Linux – et ce même si techniquement, les buts de GNU/HURD sont assez éloignés des buts de Linux.

le projet GNU, auquel il doit le compilateur GCC, qui lui a permis de développer Linux, choisit la GPL. **Il est à noter que la motivation de Torvalds à utiliser la GPL est un choix principalement pragmatique.** Quelques mois plus tard, Linux devient le noyau GNU *de facto* et apporte donc la dernière pierre à l'édifice nécessaire au projet GNU.

3. Linus le gourou ?

Linus Torvalds est souvent considéré comme l'un des gourous du logiciel libre. Certains l'élèvent même au rang de dieu du logiciel libre, une exagération qui a pour principal effet d'énerver fortement Torvalds, comme il l'explique dans son autobiographie « *Just for Fun: The Story of an Accidental Revolutionary* » (« *Rien que pour l'amusement : l'histoire d'un révolutionnaire par accident* ») (2001).

Torvalds, contrairement à Richard Stallman ou Eric Raymond (voir section suivante) n'est pas un auteur prolifique en écrits à portée philosophique (« *Talk is cheap. Show me the code.* ») (« *Parler est facile. Montrez-moi le code.* »), même s'il est célèbre pour avoir prononcé ou écrit certains « *one liners* » (courtes citations), au langage parfois fleuri, devenus des slogans du monde du logiciel libre. Si Torvalds défend bien évidemment ce dernier (« *Software is like sex: it's better when it's free* », « *Le logiciel est comme le sexe : c'est meilleur quand c'est gratuit/libre* »), ses positions sont beaucoup moins tranchées, moins extrêmes que celles de Richard Stallman.

> Ces positions peuvent se résumer par cet extrait d'une interview accordée en 1997 à Hiro Yamagata, un auteur japonais actif dans le monde du logiciel libre, interview intitulée « *The Pragmatist of Free Software: Linus Torvalds Interview*[22] » (« *Le pragmatiste du logiciel libre : Interview de Linus Torvalds* ») :
>
> « *I'm generally a very pragmatic person: that which works, works. When it comes to software, I _much_ prefer free software, because I have very seldom seen a program that has worked well enough for my needs, and having sources available can be a life-saver.*
> *So in that sense I am an avid promoter of free software, and GPL'd stuff in particular (because once it's GPL'd I _know_ it's going to stay free, so I don't have to worry about future releases)*. »

22 Yamagata, *The Pragmatist of Free Software: Linus Torvalds Interview*, 1997

(« *Je suis généralement quelqu'un de très pragmatique : ce qui fonctionne fonctionne. Quant on en vient au logiciel, je préfère _nettement_ le logiciel libre, parce que j'ai rarement vu un programme qui fonctionnait assez bien pour les besoins que j'avais et avoir accès à ses sources peut être une vraie aubaine.*
Donc, dans ce sens, je suis un avide promoteur du logiciel libre, et des trucs GPL-isés en particulier (parce qu'une fois que c'est GPL-isé, je _sais_ que ça va rester libre et je n'ai donc pas à me tracasser pour les versions suivantes). »)

Torvalds ne partage pas la haine de Richard Stallman envers le logiciel propriétaire (« *Microsoft isn't evil, they just make really crappy operating systems. », « Microsoft n'est pas maléfique, ils font seulement des systèmes d'exploitation vraiment pourris.* »). Il préfère le logiciel libre, mais reconnaît l'utilité de certains logiciels commerciaux :

« *Anybody who tells me I can't use a program because it's not open source, go suck on rms. I'm not interested. 99% of that I run tends to be open source, but that's _my_ choice, dammit.* »
(« *N'importe qui qui me dit que je ne peux pas utiliser un programme qui n'est pas open source peut aller sucer RMS. Je ne suis pas intéressé. 99% de ce que j'utilise a tendance à être open source, mais c'est _mon choix_, nom de Dieu.* »)

Il n'adhère pas aux conceptions qu'il qualifie de « religieuses » de Stallman : le concept du logiciel libre lui convient mieux par pragmatisme, parce qu'il pense que c'est un modèle supérieur au modèle fermé. Vu l'énorme succès de Linux, on comprendrait difficilement qu'il considère un autre modèle comme plus efficace.

La *dimension prophétique*[23] de Torvalds est donc à chercher sur un autre plan que chez Stallman. Ici, **point de références religieuses**, **pas de références directes à la morale**. La principale motivation de Torvalds est **son rapport ludique à la programmation**. Une autre valeur à laquelle Torvalds se dit très attaché est une certaine conception de la **paresse** :

« *Intelligence is the ability to avoid doing work, yet getting the work done.* »
(« *L'intelligence est la capacité d'éviter de travailler, tout en s'assurant que le travail est quand même fait.* »).

23 Auray, *Le prophétisme hacker et son contenu politique,* 1998

Enfin, contrairement à Stallman, Torvalds n'estime pas être parti en croisade contre les entreprises « maléfiques », mais semble penser que la supériorité du modèle libre les rendra insignifiantes, à terme :

> « *So I've never seen it as a "Linus versus Bill" thing. I just can't see myself in the position of the nemesis, since I just don't care enough. To be a nemesis, you have to actively try to destroy something, don't you? Really, I'm not out to destroy Microsoft. That will just be a completely unintentional side effect.* »
> (« *Je n'ai jamais vu ça comme un truc style « Linus contre Bill [Gates] ». Je ne me vois pas dans la position du vengeur, tout simplement parce que ça ne m'intéresse pas assez. Pour être un vengeur, vous devez essayer activement de détruire quelque chose, non ? Vraiment, je ne suis pas là pour détruire Microsoft. Ce sera juste un effet secondaire complètement non-intentionnel.* »)

3. Eric Raymond

Troisième et dernière figure de proue du logiciel libre que je citerai dans ces pages, *Eric S. Raymond* (*ESR*) est né le 4 décembre 1957. Contrairement à Stallman et Torvalds, ses contributions en termes de code informatique sont relativement faibles : son oeuvre est principalement de nature théorique et son texte « *The Cathedral and The Bazaar* » (1997) (« *La Cathédrale et le Bazar* ») est considéré comme l'un des textes fondateurs du mouvement.

Il s'agit aussi d'une personnalité de plus en plus controversée au sein de la communauté libre et *Open Source* et ses textes les plus récents sont autant de brûlots mâtinés d'un certain égocentrisme et d'opinions politiques personnelles – qu'il généralise abusivement comme représentant celle de l'ensemble des *hackers*[24], qui reprochent souvent à Raymond de profiter de son autorité pour façonner le portrait de la communauté à son image.

Les commentaires aux articles touchant de près ou de loin à la personnalité ou aux écrits de Raymond sur Slashdot, véritable baromètre des opinions de la communauté, dégénèrent presque toujours en attaques contre celui à qui ils reprochent de dénaturer la philosophie du logiciel libre pour défendre ses idées politiques ou à des fins d'auto-promotion, son égocentrisme, son révisionnisme (dans le cas du « *jargon file* ») et le titre qu'il s'attribue de

24 Jargon file, section « *politics* » : http://catb.org/~esr/jargon/html/politics.html

« *Core Linux Developer* », bien qu'aucune de ses contributions n'ait jamais été acceptée dans le noyau de Linux.

Ces précautions prises, nous pouvons analyser les apports de Raymond à la communauté *FLOSS*, apports précédant de plusieurs années les critiques sus-mentionnées.

1. « *The Cathedral and the Bazaar* » (1997)

Considéré comme un des textes fondamentaux du logiciel *FLOSS*, cet essai d'Eric Raymond décrit deux modes de développement de logiciels. Le premier, le modèle, vertical, fortement centralisé, « *built like cathedrals, carefully crafted by individual wizards or small bands of mages working in splendid isolation, with no beta to be released before its time.* » (« *construit comme des cathédrales, façonné avec soin par des sorciers isolés ou des petits groupes de mages, travaillant dans une grande isolation, où aucune version bêta n'est rendue disponible avant que son temps ne soit venu.* »). C'est le modèle typiquement utilisé par les logiciels fermés, commerciaux. On peut également remarquer ce modèle dans d'anciens logiciels libres, développés de manière très centralisée par la *Free Software Foundation*.

Dans le second modèle, dont l'auteur attribue la paternité à Linus Torvalds, le développement est largement décentralisé.

> « *Linus Torvalds's style of development-release early and often, delegate everything you can, be open to the point of promiscuity-came as a surprise. No quiet, reverent cathedral-building here-rather, the Linux community seemed to resemble a great babbling bazaar of differing agendas and approaches (aptly symbolized by the Linux archive sites, who'd take submissions from* anyone*) out of which a coherent and stable system could seemingly emerge only by a succession of miracles.* »
> (« *Le style de développement de Linus Torvalds – distribuer vite et souvent, déléguer tout ce qui est possible, être ouvert jusqu'à la promiscuité – m'apparut comme une surprise. Pas de vénération calme comme dans la construction d'une cathédrale ici – plutôt, la communauté Linux semblait ressembler à un grand bazar bourdonnant, mélangeant différents buts et approches (symbolisé très justement par les sites d'archives de Linux, qui acceptent des contributions de* n'importe qui*), hors desquels un système cohérent et stable ne pourrait apparemment émerger uniquement par une succession de miracles.* »).

La grande nouveauté du système « bazar », tel que décrit par Raymond est ce qu'il appelle « la Loi de Linus »: « *Given enough eyeballs, all bugs are shallow* » (« *Étant donné un nombre suffisant d'yeux, tous les bugs sont superficiels* »). Ce slogan fait référence au fait qu'un grand nombre de développeurs, disséminés et d'horizon différents aura plus de chances de trouver rapidement des solutions à des bugs qu'un petit groupe de développeurs spécialisés, typiques du modèle « bazar ». Il s'agit, en d'autres termes, de la victoire du système de revue des pairs sur un système basé sur le secret et l'obscurité.

Notons également que « *The Cathedral and The Bazaar* » fut suivi de deux documents, moins fondamentaux, que je me contenterai de citer : « *Homesteading the Noosphere* » (« *À la conquête de la noosphère* »), en 1998 et « *The Magic Cauldron* » (« *Le Chaudron Magique* »), en 1999. Cette trilogie fut ensuite compilée, révisée et imprimée aux éditions O'-Reilly sous le titre générique « *The Cathedral and The Bazaar* ». « *Homesteading the Noosphere* » est l'occasion pour Raymond de s'essayer à un texte sur l'anthropologie du don, thème que je ne ferai qu'effleurer dans ce texte, pour des contraintes de format.

2. L'« *Open Source Initiative* » (1998)

En 1998, Eric Raymond fonda l'« *Open Source Initiative* » (« *Initiative Open Source* », *OSI*), une organisation destinée à promouvoir le logiciel *Open Source*. C'est à cette occasion que le vocable « *Open Source* » fut inventé. Le but de cette organisation était de se démarquer du logiciel libre, en lui enlevant ses caractéristiques morales et idéologiques, dans le but d'être plus compatible avec les entreprises, rendues frileuses par le caractère « viral » de la GPL – par son *copyleft*.

La définition du logiciel *Open Source* est cependant à peu près semblable à celle de la définition du logiciel libre et inclut donc, par exemple, les logiciels sous licence GPL ou BSD.

Eric Raymond s'est récemment rétracté de sa position de président de l'OSI, en février 2005. À ce jour, le bilan de l'organisation reste mitigé : elle aura donné une meilleure visibilité au logiciel *Open Source* (et donc au logiciel libre), mais aura aussi créé un schisme au sein de la communauté *FLOSS*. Elle aura également contribué à entretenir une certaine ambiguïté aux yeux du grand public entre le logiciel libre, le logiciel *Open Source* et le logiciel sous GPL.

Partie II
Références théoriques

I. Pekka Himanen

Pekka Himanen (né le 19 novembre 1973) est un philosophe finlandais, auteur de « *The Hacker Ethic and the Spirit of the Information Age* » (2001). Aujourd'hui, il est le directeur du « *Berkeley Center for Information Society* » (Université de Berkeley, Californie). Il est également chercheur à l'Université d'Helsinki. Il est à noter qu'Himanen n'est pas sociologue et que « *The Hacker Ethic and the Spirit of the Information Age* » est un ouvrage de philosophie, non de sociologie.

Les deux premières parties de ce livre, celles qui nous préoccuperont comparent l'« *éthique hacker* » et l'éthique protestante telle que décrite par Weber dans son ouvrage « *L'éthique protestante et l'esprit du capitalisme* ». Ouvrage duquel je vais rappeler quelques notions fondamentales à la section suivante. J'attire l'attention du lecteur qu'il ne s'agit pas d'un résumé de l'oeuvre de Weber – tel n'est pas le propos de ce travail –, mais d'un bref aperçu des quelques notions nécessaires aux comparaisons qu'Himanen propose dans son ouvrage.

La théorie bien connue de Weber est que le capitalisme se serait développé sous l'influence du calvinisme :

> « *L'un des éléments fondamentaux de l'esprit du capitalisme moderne, et non seulement de celui-ci, mais de la civilisation moderne elle-même, à savoir : la conduite rationnelle fondée sur l'idée de* Beruf*, est né de l'esprit de l'ascétisme chrétien.* » (Max Weber, « *L'éthique protestante et l'esprit du capitalisme* », 2002, p. 140).

II. Max Weber – quelques notions issues de la théorie de « *L'éthique protestante et l'esprit du capitalisme* »

L'Église catholique romaine promettait le salut à tous, contre quelques indulgences :

> « *Mais les sceptiques et les indifférents eux-mêmes se réconciliaient avec l'Église par des dons : en tout état de cause, il valait mieux s'assurer contre l'incertitude de ce qui pouvait se passer après la mort, et (du moins selon une conception assez relâchée bien qu'extrêmement répandue) une soumission extérieure aux commandements de l'Église suffisait à assurer le salut.* » (Max Weber, 2002, p. 42).

La Réforme supprima ces indulgences et laissa le réformé calviniste sans moyen de s'assurer qu'il est élu, qu'il possède son entrée au paradis après sa mort (la « *possessio salutis* »).

D'autre part, le réformé est convaincu que l'Homme est prédestiné :

> « *Admettre que le mérite ou la culpabilité des humains ait une part quelconque dans la détermination de leur destin reviendrait à considérer que les décrets absolument libres de Dieu, et pris de toute éternité, puissent être modifiés sous l'influence humaine – pensée qu'il n'est pas possible de concevoir.* » (Max Weber, 2002, p. 67).

Il n'est donc pas question, pour lui, d'oeuvrer de manière à gagner son salut (et donc, *a fortiori*, de pouvoir acheter celui-ci).

Ainsi, *le réformé vit dans une profonde angoisse* (Olgierd Kuty, 1998, p. 103). Ce n'est que par des moyens détournés qu'il lui est possible de trouver des **signes** de son élection.

> « *Autant les bonnes oeuvres sont absolument impropres comme moyen pour obtenir le salut – l'élu lui-même restant une créature, tout ce qu'il fait est infiniment éloigné de ce que Dieu exige –, autant elles demeurent indispensables comme signes d'élection. Moyen technique, non pas sans doute d'acheter le salut, mais de se délivrer de l'angoisse du salut. En ce sens, on dira, à l'occasion, qu'elles sont directement « indispensables au salut », ou bien que la* possessio salutis *en dépend.*
> *Dans la pratique, cela signifie que Dieu vient en aide à qui s'aide lui-même ; et aussi*

que le calviniste, comme on l'a dit parfois, « crée » lui-même son propre salut. Ou, plus correctement, la certitude de celui-ci. » (Max Weber, 2002, pp. 78-79)

C'est donc dans ses « bonnes oeuvres » que le réformé va rechercher ces signes. Et la nature de ces bonnes oeuvres est stricte :

« *Le Dieu du calvinisme réclamait non pas des bonnes oeuvres isolées, mais une vie tout entière de bonnes oeuvres érigées en système. Pas question du va-et-vient catholique, authentiquement humain, entre péché, repentir, pénitence, absolution, suivis derechef du péché.* » (Max Weber, 2002, p. 81).

Le réformé calviniste va donc ériger un système de « bonnes oeuvres », et pour ce faire transférer dans le siècle la discipline ascétique des moines, en ce compris leur division du temps :

« [N]*ous ne devons pas oublier qu'au Moyen Age les moines furent les premiers à vivre selon une minutieuse division du temps, et qu'en cela surtout résidait l'utilité des cloches.* » (Max Weber, 2002, p. 117)

Et comment définir « bonne oeuvre » ? La réponse se trouve dans la Bible : par le travail :

« *[S]ur terre, l'homme doit, pour assurer son salut, « faire la besogne de Celui qui l'a envoyé, aussi longtemps que dure le jour » [Jean IX, 4]. Ce n'est ni l'oisiveté ni la jouissance, mais l'activité seule qui sert à accroître la gloire de Dieu, selon les manifestations sans équivoque de sa volonté.* » (Max Weber, 2002, p. 116)

Le travail acquiert donc une valeur, une légitimité intrinsèques.

« *L'individu doit exceller dans un métier, qui doit être compris comme un don, voire même comme une vocation. S'il en est bien ainsi, choisir son métier revient en toute logique à choisir d'exercer le métier pour lequel des dons nous ont été donnés par Dieu. (...) Chaque métier est d'ailleurs également estimable, du plus manuel au plus intellectuel. Dans cette perspective, la reconnaissance qu'un individu reçoit de la part des autres dépend avant tout de son assiduité, de sa persévérance, de son courage,*

voire de son obstination à faire correctement son métier, quel qu'il soit. (...) L'assiduité au travail produit naturellement de l'argent. Comme le métier, détenir des gains légitimes peut être un autre signe annonçant la salvation *après la mort. Prêter de l'argent à un autre membre de sa communauté, l'utiliser de cette façon pour procurer du travail à un autre, c'est à la fois faire prospérer ses propres affaires mais aussi oeuvrer au bien commun et, par extension, travailler à la gloire de Dieu sur terre.* »
(Claude Macquet, Didier Vrancken, 2003, p. 81)

Comme on peut le voir dans ce résumé de quelques lignes, selon Weber, l'anxiété des réformés calvinistes, à la recherche d'un signe de leur élection, aura entraîné une valorisation intrinsèque du travail, de l'argent et aura donc entraîné la naissance de l'« esprit du capitalisme ».

III. Mise en rapport de l'éthique protestante et de l'*éthique hacker*

« *Hacker ethic : The belief that information-sharing is a powerful positive good, and that it is an ethical duty of hackers to share their expertise by writing open-source code and facilitating access to information and to computing resources wherever possible.*[25] »
(« *L'éthique hacker : La croyance que le partage de l'information est un bien positif et que les hackers ont le devoir éthique de partager leur expertise en écrivant du code open-source et en facilitant l'accès à l'information et aux ressources informatiques quand cela est possible.* »)

1. L'académie et le monastère

Une des contributions majeures de l'ouvrage, à mon sens, est de proposer une meilleure métaphore pour différencier les modèles qu'Eric Raymond appelait « Cathédrale » et « Bazar ». Chez Himanen, ces modèles sont désignés comme, respectivement, le « monastère » et l'« académie ».

25 Jargon file, section « *hacker ethic* » : http://catb.org/~esr/jargon/html/H/hacker-ethic.html

	Monastère	***Académie***
Équivalent chez Raymond	Cathédrale	Bazar
Modèle de développement	Système « fermé », *closed source*, centralisation forte, hiérarchie forte.	Système ouvert, *Open Source*, décentralisation[26]. La hiérarchisation varie selon les projets.
Relecture	Par un petit groupe fermé, sans intervention de programmeurs extérieurs au projet.	Revue des pairs, *loi de Linus* : « *Given enough eyeballs, all bugs are shallow* » (« *Étant donné un nombre suffisant d'yeux, tous les bugs sont superficiels* »)
Éthique	Éthique protestante	*Éthique hacker*
Ère	Ère capitaliste	Ère de l'information

Le monastère, bien entendu, fait directement référence au monastère sécularisé de Weber, au modèle de fonctionnement des entreprises capitalistes :

> « *The opposite of this hacker and academic open model can be called the closed model, which does not just close off information but is also authoritarian. In a business enterprise built on the monastery model, authority sets the goal and chooses a closed group of people to implement it. After the group has completed its own testing, others will have to accept the result purely as it is. Other uses of it are called "unauthorized uses." We can use our allegory of the monastery as an apt metaphor for this style, which is well summed up by Saint Basil the Great's monastic rule from the fourth century: "No one is to concern himself with the superior's method of administration or make curious inquiries about what is being done." The closed model does not allow for initiative or criticism that would enable an activity to become more creative and self-corrective.* » (Himanen, 2001, p. 70)

26 Cette décentralisation est l'un des phénomènes qui permet la réfutation du terme « communisme » parfois employé à l'encontre du mouvement par ses détracteurs. « *[T]andis que le communisme renvoie souvent à une structure autoritaire centralisée, la thèse fait de l'Académie un espace de dialogue et d'ouverture. Himanen critique le choix par Merton d'utiliser le label de communisme pour qualifier l'éthique scientifique.* » (Auray, *De l'éthique à la politique : l'institution d'une cité libre*, 2002).
Notons que l'emploi du terme « communisme scientifique » par Merton ne fait pas référence au système politique, mais simplement à une mise en commun des connaissances, raison pour laquelle Himanen critique ce choix lexical (Himanen, *The Hacker Ethic*, 2001, p. 60)

(« *L'opposé de ce modèle ouvert, hacker et académique, peut aussi être appelé le modèle fermé ; il ne se contente pas de renfermer l'information, mais est aussi autoritaire. Dans une entreprise commerciale construite sur le modèle du monastère, l'autorité définit un but et nomme un groupe fermé de personnes pour l'implémenter. Une fois que le groupe a terminé ses propres tests, d'autres devront accepter le résultat purement comme il est. Les autres utilisations du produit sont appelées les « utilisations non-autorisées ». Nous pouvons utiliser notre allégorie du monastère comme une métaphore valable pour ce style, qui est bien résumé par la règle monastique de Saint Basile le Grand, au IVème siècle : « Personne ne doit s'immiscer dans les méthodes d'administration de son supérieur ni manifester de curiosité au sujet de ce qui est fait. » Le modèle fermé ne permet pas l'initiative ou la critique qui permettraient à une activité de devenir plus créative et auto-correctrice.* »)

L'originalité de l'analyse d'Himanen se trouve dans le modèle qu'il oppose au modèle du monastère, à savoir le modèle de l'académie platonicienne. Dans un passage particulièrement éclairant, Himanen explique :

« *Scientists, too, release their work openly to others for their use, testing, and further development. Their research is based on the idea of an open and self-correcting process. The sociologist Robert Merton wrote that this idea of self-correction was as important a principle to science as openness. He called it* organized skepticism– *historically, it is a continuation of the* synusia *of Plato's Academy, which also included the idea of approaching the truth through critical dialogue. The scientific ethic entails a model in which theories are developed collectively and their flaws are perceived and gradually removed by means of criticism provided by the entire scientific community.*

Of course, scientists have chosen this model not only for ethical reasons but also because it has proved to be the most successful way of creating scientific knowledge. All of our understanding of nature is based on this academic or scientific model. The reason why the hackers' open-source model works so effectively seems to be–in addition to the facts that they are realizing their passions and are motivated by peer recognition, as scientists are, too–that to a great degree it conforms to the ideal open academic model, which is historically the best adapted for information creation. » (Himanen, 2001, pp. 68-69)

(« *Les scientifiques aussi publient leur travail ouvertement pour que d'autres l'utilisent, le testent et le développent plus avant. Leur recherche est basée sur l'idée d'un processus ouvert et auto-correcteur. Le sociologue Robert Merton écrivit que cette idée de l'auto-correction est un principe aussi important à la science que l'ouverture. Il le nomma* scepticisme organisé *– historiquement, c'est la continuation de la* synusia *de l'Académie de Platon, qui incluait aussi l'idée d'approcher la vérité par l'intermédiaire du dialogue critique. L'éthique scientifique implique un modèle dans lequel les théories sont développées collectivement et leurs fautes sont perçues et graduellement enlevées par le fait de la critique fournie par toute la communauté scientifique.*
Bien sûr, ***les scientifiques ont choisi ce modèle*** *non seulement pour des raisons éthiques mais aussi* ***parce qu'il a fait ses preuves comme la meilleure manière de créer la connaissance scientifique****.* ***La raison pour laquelle le modèle Open Source des hackers fonctionne de manière si efficace semble être*** *– en plus des faits qu'ils réalisent leurs passions et sont motivés par la reconnaissances des pairs, comme le sont aussi les scientifiques –* ***que, dans une grande mesure, il se conforme au modèle ouvert académique idéal, qui est, historiquement, le mieux adapté pour la création de l'information****.* »)

Voici une clef capitale à la compréhension du mouvement, de son succès, l'explication du sous-titre du présent travail : l'*innovation sociale* proposée par le logiciel libre est d'abord caractérisée par la rupture avec le modèle, secret et fermé, de développement du logiciel fermé, au profit d'une grande transparence. L'intuition d'Himanen, qui avance que **la raison du succès du logiciel libre est qu'il constitue un moyen mieux adapté à la création de l'information** et rappelle que la science a toujours fonctionné selon ce modèle, est séduisante. Sans me prononcer sur la validité de cette intuition, j'attirerai l'attention du lecteur sur le fait que **les communautés de développement libre** – et nous en verrons un exemple plus loin dans mon analyse empirique – **présentent un grand attachement à la croyance en la valeur de la méthode scientifique**.

Les trois autres apports importants de l'analyse d'Himanen sur l'*éthique hacker* sont ce qu'il a appelé l'éthique de travail et l'éthique de l'argent. Entre ces deux sujets, j'évoquerai le rapport du *hacker* au temps, selon Himanen.

2. L'éthique du travail

Nous avons vu que chez Weber, l'éthique protestante impose à l'individu d'exceller dans un métier, qui doit être compris comme un don, une vocation (*Beruf*). La nature du travail pratiqué n'importe pas, tout travail étant également estimable : comme dans l'adage « il n'est point de sot métier », le travail a une valeur intrinsèque et l'important est d'accomplir ce travail tant que dure le jour, parce que c'est ainsi que Dieu l'a voulu.

Chez le *hacker* d'Himanen, il en va tout autrement. Himanen cite cet extrait d'Eric Raymond :

> « *To do the Unix philosophy right, you have to be loyal to excellence. You have to believe that software design is a craft worth all the intelligence and passion you can muster...*
> *Software design and implementation should be a joyous art, a kind of high-level play. If this attitude seems preposterous or vaguely embarrassing to you, stop and think; ask yourself what you've forgotten. Why do you design software instead of doing something else to make money or pass the time? You must have thought software was worthy of your passion once...*
> *To do the Unix philosophy right, you need to have (or recover) that attitude. You need to care. You need to play. You need to be willing to explore.* » (Eric Raymond, « *The Art of Unix Programming* » (2001), cité dans Himanen, 2001, pp. 5 et 6)
> (« *Pour appliquer correctement la philosophie Unix, tu dois être d'une grande loyauté. Tu dois croire que la conception d'un logiciel est un art qui mérite toute l'intelligence et la passion que tu peux réunir...* »
> *La conception et l'implémentation d'un logiciel doit être un art jubilatoire, un genre de jeu de haut niveau. Si cette attitude te semble absurde ou quelque peu embarrassante, arrête-toi et réfléchis ; demande-toi ce que tu as oublié. Pourquoi conçois-tu des logiciels au lieu de faire quelque chose d'autre pour te faire de l'argent ou pour passer le temps ? Tu dois bien avoir pensé un jour que le logiciel était ta passion...*
> *Pour suivre correctement la philosophie Unix, tu dois avoir (ou retrouver) cette attitude. Tu dois avoir du soin. Tu dois jouer. Tu dois avoir envie d'explorer.* »)

Le rapport au travail est ici un rapport ludique, attentionné et passionné. Loin d'être une fin en soi, **le travail est ici un art jubilatoire**, un jeu de haut niveau. Le principal moteur du *hacker* est sa **passion**.

> « *[The hacker ethic] resembles the pre-Protestant ethic to a much greater degree than it does the Protestant one. (...) But, it is important to note, only closer: ultimately, the hacker ethic is not the same as the pre-Protestant work ethic, which envisions an attainable paradise of life without doing anything. Hackers want to realize their passions, and they are ready to accept that the pursuit even of interesting tasks may not always be unmitigated bliss.* » (Himanen, 2001, p. 18).
> (« *L'éthique hacker ressemble à l'éthique pré-protestante beaucoup plus qu'elle ne ressemble à l'éthique protestante. (...) Mais, il est important de le noter, elle en est seulement plus proche : en définitive, l'éthique hacker n'est pas celle de l'éthique de travail pré-protestante, qui envisageait un paradis accessible sans avoir à faire quoi que ce soit. Les hackers veulent réaliser leurs passions, et ils sont prêts à accepter que la poursuite même des tâches intéressantes peut ne pas toujours être un bonheur sans nuance.* »)

Himanen continue en citant Linus Torvalds : « *Linux has very much been a hobby (but a serious one: the best type.* » (Himanen, 2001, pp. 18-19) (« *Linux a toujours était surtout un hobby (mais un hobby sérieux : le meilleur type)*. »). Si le rapport au travail a changé, si le travail a perdu la valeur intrinsèque qu'il avait chez Weber, il n'en reste pas moins une chose sérieuse et intense. Ce qui a changé fondamentalement, c'est la **motivation** du *hacker* : il n'est plus question d'obligation divine, de vocation (*Beruf*), mais de réalisation d'une vie passionnée.

3. Rapport du *hacker* au temps : « le temps flexible »

Lorsque Weber cite Benjamin Franklin au début de *L'éthique protestante et l'esprit du capitalisme*, il introduit l'adage « *Time is money* » (« *Le temps, c'est de l'argent* »). L'éthique protestante, chez Weber, demande une gestion rigoureuse du temps et érige la régularité en principe. Comme nous l'avons vu plus tôt, pour Weber, l'organisation du temps capitaliste descend en droite ligne de l'organisation monacale, dont elle constitue la sécularisation.

Himanen introduit la notion de « *flexible time* » (Himanen, 2001, p. 29) (« ***temps flexible*** ») : le *hacker* suit le rythme de sa créativité et **dispose de son temps**. Travail, famille, amis, hobbies sont entremêlés, la démarcation entre ces différents segments tend à disparaître. L'adage « *time is money* » (« *le temps, c'est de l'argent* ») est remplacé par « *it's my life* » (« *c'est ma vie* »).

> « *The hacker ethic also reminds us, in the midst of all the curtailment of individual worth and freedom that goes on in the name of "work", that our life is here and now. Work is a part of our continuously ongoing life, in which there must be room for other passions, too. Reforming the forms of work is a matter not only of respecting the workers but of respecting human beings as human beings. Hackers do not subscribe to the adage "time is money" but rather to the adage "it's my life". And certainly this is now our life, which we must live fully, not a stripped beta version of it.* » (Himanen, 2001, p. 40)
>
> (« *L'éthique hacker nous rappelle aussi, au milieu de tout le rétrécissement de la valeur individuelle et de la liberté qui vont de pair au nom du « travail », que notre vie est ici et maintenant. Le travail est une partie de notre vie en mouvement perpétuel, dans laquelle il doit y avoir de la place pour d'autres passions, aussi. Réformer les formes de travail est une manière non seulement de respecter les travailleurs mais aussi de respecter les êtres humains en tant qu'êtres humains. Les hackers ne souscrivent pas à l'adage « le temps, c'est de l'argent », mais plutôt à l'adage « c'est ma vie ». Et très certainement, c'est maintenant notre vie, que nous devons vivre pleinement, et non une version bêta dépouillée de celle-ci.* »)

Il est important de noter que ce nouveau rapport au temps n'est pas, dans l'*éthique hacker*, la manifestation d'un manque de sérieux, de fainéantise ou de paresse. Tout au contraire, il s'agit d'organiser son temps de manière plus flexible[27]. Dans cette optique, suivre le rythme de sa créativité peut être vu comme un moyen d'**optimiser son travail**. Le but n'est plus de travailler de 9 à 18 tous les jours, mais que le travail soit fait (« *to get the job done* »).

---

27 « *[D]ans la version* hacker *du temps libéré, différentes séquences de la vie comme le travail, la famille, les amis, les hobbies, sont mélangés avec une souplesse de telle sorte que le travail n'occupe jamais le centre.* » (Auray, *De l'éthique à la politique : l'institution d'une cité libre*, 2002)

4. L'éthique de l'argent

Chez Weber, l'argent apparaît comme le corollaire du travail. Travail et argent sont considérés comme des fins en eux-mêmes (Himanen, 2001, p. 44).

> « *L'assiduité au travail produit naturellement de l'argent. Comme le métier, détenir des gains légitimes peut être un autre signe annonçant la* salvation *après la mort.* » (Claude Macquet, Didier Vrancken, 2003, p. 81).

Dans le cadre de l'*éthique hacker*, par contre, il est difficilement admissible de tirer profit de l'information en maintenant celle-ci secrète. Toutefois, il est paraît important de noter ici que le *hacker* n'a pas fait voeu de pauvreté. Ce n'est pas l'argent qu'il répugne, c'est **le fait de refuser de partager l'information dans le but de gagner de l'argent** :

> « *That said, one should not think of most hackers' attitude toward money as either some paradisiacal utopianism or some kind of aversion to it. (...) Hackers are not naïve. They are not blind to the fact that in a capitalist society it is actually very difficult to be completely free unless a person has sufficient capital.* » (Himanen, 2001, p. 53-54)
>
> (« *Ceci dit, on ne doit pas penser l'attitude de la plupart des hackers envers l'argent ni comme un quelconque utopisme paradisiaque, ni comme un genre d'aversion envers lui. (...) Les hackers ne sont pas naïfs. Ils ne sont pas aveugles au fait que dans une société capitaliste, il est en fait très difficile d'être complètement libre, à moins de disposer d'un capital suffisant.* »)

C'est en cherchant à créer un nouveau modèle économique que le *hacker* arrive à concilier passion et capitalisme. Cette nouvelle voie, ils la trouvent dans la notion de service, qu'ils substituent à la notion de produit. (Himanen, 2001, p. 57). Ainsi, le logiciel est développé de manière libre (Himanen cite le cas de la société Red Hat, spécialisée dans une distribution de Linux portant ce même nom), mais ces nouvelles entreprises vendent un panel de services pour accompagner ledit produit.

5. Reconnaissance par la communauté

S'il y a un point où le capitaliste weberien et le *hacker* se rejoignent, c'est dans la **recherche d'une reconnaissance de la part de la communauté**. Cette reconnaissance est atteinte, chez Weber, par le travail et l'argent. Chez Himanen en revanche, cette reconnaissance est celle, directe, des pairs. Nous l'avons vu, le *hacker* cherche à réaliser son travail comme un art, avec passion :

> « *For these hackers, recognition within a community that shares their passion is more important and more deeply satisfying than money, just as it is for scholars in academe. The decisive difference from the Protestant ethic is that for hackers it is important that peer recognition is no substitute for passion - it must come as a* result *of passionate action, of the creation of something socially valuable to this creative community. Under the Protestant ethic, the opposite is often the case: social motivations serve to distract attention from the idea that work itself should involve the realization of a passion. As a result, the Protestant ethic's emphasis on the social features of work becomes a double surrogate: for the lack of social life outside of work* and *for the absence of an element of passion in the work itself.* » (Himanen, 2001, p. 51)
> (« *Pour ces hackers, la reconnaissance dans une communauté que partage leurs passions est plus importante est plus profondément satisfaisante que l'argent, exactement comme ce l'est pour les disciples de l'académie. La différence décisive par rapport à l'éthique protestante est que pour les hackers, il est important que la reconnaissance des pairs n'est pas un substitut à la passion – elle doit apparaître comme le* résultat *de l'action passionnée, de la création de quelque chose de socialement utile pour cette communauté créative. Sous l'éthique protestante, l'inverse est souvent le cas : les motivations sociales servent à distraire l'attention de l'idée que le travail lui-même doit impliquer la réalisation d'une passion. Comme résultat, l'emphase de l'éthique protestante sur les caractéristiques sociales du travail devient un double substitut : pour le manque de vie sociale en-dehors du travail* et *pour l'absence d'un élément de passion dans le travail lui-même.* »)

La mesure principale de la qualité du travail ne sera donc plus l'argent chez le *hacker*, mais le fait que sa création ait une valeur sociale.

6. Les valeurs protestantes *vs* les valeurs *hackers*, chez Himanen

Himanen conclut son ouvrage en relevant les sept valeurs de l'éthique protestante, qu'il compare aux sept valeurs de l'éthique *hacker*. En voici un tableau récapitulatif, en guise de conclusion à l'analyse d'Himanen.

	Éthique protestante	***Éthique hacker***
Valeurs premières	Le travail, l'argent	La créativité, la passion
Autres valeurs	L'optimalité, la flexibilité, la stabilité, la détermination et la responsabilité des résultats.	La liberté, la valeur sociale de la création, la dimension d'ouverture, l'activité, l'attention à l'égard des autres

7. Conclusions sur l'oeuvre d'Himanen

Voici résumées les deux premières parties de l'ouvrage d'Himanen ainsi qu'un bref aspect de sa conclusion. Je n'aborderai pas en ces pages la troisième partie du livre, ni les autres éléments de sa conclusion, qui me paraissent moins pertinents quant à mon sujet d'étude et aux ambitions sociologiques de ce travail. En effet, ces pages s'éloignent de l'analyse des communautés de *hackers* en tant que telles et proposent un texte peut-être plus philosophique, moins cadré, parfois mystique, parfois religieux – l'auteur aborde le thème du développement personnel, parodie la Genèse revue par l'éthique protestante.

Notons également que certaines notions abordées dans les deux premières parties du livre, comme celles du « temps flexible » et de l'« éthique de l'argent », malgré leur pertinence dans leur comparaison avec Weber, sont parfois marquées de parti-pris de la part de l'auteur, sinon d'utopie – même s'il s'en défend ouvertement[28].

Dans mon analyse empirique d'une communauté de développeurs libres, dans la dernière partie de ce travail, je reviendrai sur les thèmes du rapport au temps et à l'argent des

28 Auray aussi se montre critique quant à l'échec de l'ouvrage de proposer un modèle politique cohérent avec les idées « *hackers* » d'Himanen : « *C'est précisément toute la faiblesse de l'ouvrage que de ne jamais proposer une conception véritablement politique dans la perspective d'une transformation de la société de l'information. Des deux points particulièrement faibles soulevés par une telle perspective éthique, le premier concerne la liberté. Quel est l'horizon de liberté que dessine une telle éthique ? Que faire de la liberté telle qu'elle est soulevée par Himanen ?* » (Auray, *De l'éthique à la politique : l'institution d'une cité libre*, 2002)

communautés de développeurs libres. Certaines de mes conclusions prendront quelques distances par rapport à la vision à mon sens parfois trop poétique qu'a Himanen de son *hacker*.

Partie III
Analyse empirique

I. Des pommes, des poires et des... gnous

Parmi les centaines de projets de logiciel libre actifs, j'ai dû en choisir un pour l'analyser par l'intermédiaire de sa mailing-list. Mon choix s'est porté sur un logiciel assez récent, PearPC. J'aurais pu choisir d'analyser différents projets logiciels développés selon la licence du projet GNU, la GPL (ou même selon une autre licence libre) mais, à l'heure d'arrêter une décision, c'est sur la mailing-list de PearPC que mon choix s'est porté. Ce projet est né, publiquement du moins, le 4 mai 2004 à 14h13, date de son apparition sur SourceForge.net, le site qui fédère un grand nombre de projets libres.

Plusieurs raisons ont motivé mon choix :

- J'étais personnellement un habitué de cette liste, dès ses premiers jours (en fait plus un *lurker*[29]) ;
- Sur un peu plus d'un an d'activité, de nombreux événements que je pourrais caractériser de « typiques » dans ce type de communauté sont apparus ;
- Le niveau technique y est excellent ;
- Il s'agit d'un projet de taille modeste, ce qui me permet d'appréhender les messages postés sur sa mailing-list dans leur intégralité[30].

---

29 « *lurker: n. One of the 'silent majority' in an electronic forum; one who posts occasionally or not at all but is known to read the group's postings regularly. This term is not pejorative and indeed is casually used reflexively: "Oh, I'm just lurking." (...)* » (Jargon File, section « *lurker* », http://www.catb.org/~esr/jargon/html/L/lurker.html)
(« *Lurker : n. Un membre de la « majorité silencieuse » dans un forum électronique ; quelqu'un qui poste occasionnellement ou pas du tout, mais dont on sait qu'il lit les messages du groupe régulièrement. Ce terme n'est pas péjoratif et en fait utilisé de manière réflexive : « Oh, je ne fais que lurker.* »)

30 Notons qu'il n'existe pas de moyen facile pour connaître le nombre d'abonnés à PearPC-devel. Il est possible de compter le nombre d'*intervenants* depuis la création de la liste, en mai 2004 (plus ou moins de 400 intervenants au total, dont 20 ou 30 postent régulièrement sur la liste), mais ce nombre ne renseigne pas sur le nombre de *lurkers* inscrits à la liste.
Il est également à noter que, sur ces 400 intervenants, le nombre de femmes est étonnement faible : j'en ai compté 2 ou 3, en tout et pour tout : moins de 1% de femmes constituent le groupe des intervenants sur la liste.

L'un des buts de ce document est d'éviter d'entrer dans des considérations techniques, qui n'apporteraient rien – ou si peu – à ma réflexion et décourageraient le lecteur non-technicien. Ma présentation du projet PearPC se limitera donc aux quelques lignes suivantes, où j'essayerai de décrire les buts du projet précisément et sans technicité inutile.

Ces précautions prises, il me faudra mentionner que PearPC est un logiciel dit « émulateur ». Un émulateur est un logiciel permettant de faire fonctionner sur un ordinateur d'une architecture X, des programmes écrits pour une architecture Y. Les ordinateurs n'étant que des machines de Turing, c'est-à-dire des dispositifs maniant uniquement des séquences gigantesques de 0 et de 1, l'émulation complète et parfaite d'une machine Y par une machine X est théoriquement possible. Cependant, l'étape de « traduction simultanée » imposée par l'émulateur introduit un bémol : celle-ci ne permet de fournir qu'une machine émulée (le client) aux performances 20 fois moindres par rapport aux performances de la machine « émulatrice » (l'hôte).

Plusieurs méthodes peuvent permettre d'arriver à cette fin, que l'on pourrait présenter sur une palette allant de la simple simulation à la recréation pure de la machine à imiter, dans ses détails les plus intimes (Fig. 4). Dans le premier cas, seules certaines fonctionnalités de l'ordinateur à recréer sont disponibles, souvent dans le but très pragmatique de faire tourner tel ou tel programme dans certaines conditions strictes. Dans le second cas, l'ordinateur à recréer l'est avec une telle fidélité que celui-ci, pour employer une métaphore anthropomorphiste, « pense » être l'ordinateur cible et agit en tout point comme la machine imitée. Un émulateur de ce type ne fonctionnera donc pas différemment de l'ordinateur imité.

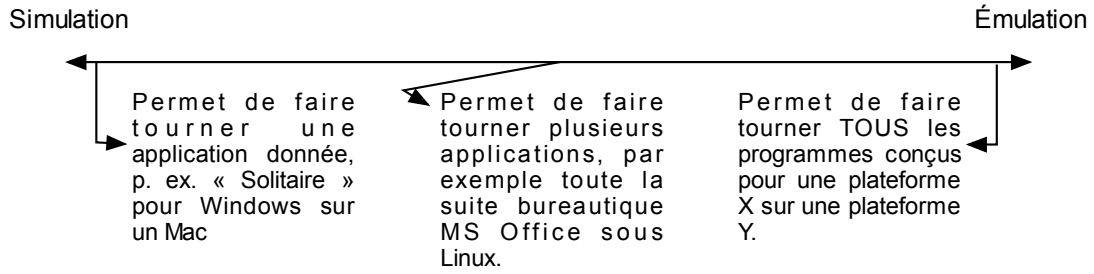


Fig. 4 : de simulation à émulation

Si la simulation est déjà un tour de force, l'émulation, elle, est une vraie épreuve de discipline et de précision, d'autant plus que les spécifications de la machine à imiter ne sont jamais disponibles dans une documentation exhaustive : certains composants d'une machine peuvent être très bien documentés (un microprocesseur, par exemple, tout complexe qu'il soit,

dispose normalement d'une documentation exhaustive), tandis que d'autres ne le sont pas, soit parce que cette documentation n'intéresserait théoriquement que les créateurs de semi-conducteurs et n'est donc publiée nulle part, soit parce que cette information n'est pas divulguée par le constructeur du composant, qui estime qu'il s'agit d'un secret de fabrique.

Dès le départ, les créateurs de PearPC ont décidé de se situer le plus à droite possible sur l'axe représenté en Fig. 4. Le but officiel de PearPC est en effet de documenter parfaitement tous les aspects techniques d'un ordinateur Apple Macintosh. Que ce logiciel permette d'exécuter les programmes prévus pour ce derniers sur PC n'est, en définitive, qu'un effet secondaire positif, aux yeux des initiateurs du projet :

> (« PearPC is what the first sentense on the homepage tells: a PowerPC Architecture Emulator. It is not a Mac OS X compatibility layer like many people want ») (*Sebastian Biallas*, 2 juillet 2004, 16:19)
>
> (« PearPC est ce que la première phrase de la homepage dit : un émulateur d'architecture PowerPC. Ce n'est pas une couche de compatibilité pour Mac OS X comme le voudrait beaucoup de monde. »)

Le but est d'**accumuler de la connaissance et du savoir**. Ce sujet, on l'aura compris, sera à l'origine de nombreuses disputes sur la mailing-list entre ce que j'appellerai les « *pragmatistes* » et les « *architectes* », disputes que nous analyserons plus loin dans ce document.

Il ne me reste plus qu'à expliquer le nom du logiciel, PearPC, qui est en fait un double jeu de mots : le microprocesseur qui équipe aujourd'hui les ordinateurs Apple Macintosh[31] est connu sous le nom de « PowerPC », généralement abrégé en « PPC ». PPC se prononce, en anglais, d'une manière proche de « Pear - PC ». L'émulateur visant à recréer un ordinateur de marque Apple (« pomme »), le terme pear (« poire ») était tout indiqué.

31 Le 6 juin 2005, Apple a annoncé son intention d'abandonner l'architecture PowerPC au profit de l'architecture Intel x86 dans les 12 à 18 mois, ce qui aura, à l'avenir, plus que certainement des conséquences sur l'évolution de PearPC. Toutefois, la très grande partie des messages analysés dans ce travail est antérieure à cette annonce et n'a donc aucun impact sur ceux-ci.

II. La hiérarchie de la Poire

Ces précisions apportées, oublions-les et revenons à notre mailing-list, ou plutôt, devrais-je dire, à nos mailing-lists : il existe en effet deux mailing-lists PearPC, *PearPC-devel* et *PearPC-users*. Comme leurs noms respectifs l'impliquent, la première serait réservée aux développeurs et la seconde aux « simples » utilisateurs. Un bref coup d'oeil sur le nombre de messages (plus de 5 000 sur PearPC-devel depuis la création du projet, moins de 500 sur PearPC-users, voir Fig. 5) nous informe déjà que les choses ne sont pas aussi simples[32].

PearPC-devel, répartition des « posts » du 2 mai 2004 au 14 août 2005 (5339 au total) :

	Jan.	***Févr.***	***Mars***	***Avril***	***Mai***	***Juin***	***Juil.***	***Août***	***Sep.***	***Oct.***	***Nov.***	***Déc.***
2004					715	846	582	344	206	475	189	80
2005	146	85	476	757	152	141	119	26				

PearPC-users, répartition des « posts » du 2 mai 2004 au 14 août 2005 (417 au total) :

	Jan.	***Févr.***	***Mars***	***Avril***	***Mai***	***Juin***	***Juil.***	***Août***	***Sep.***	***Oct.***	***Nov.***	***Déc.***
2004					77	62	38	22	16	23	29	3
2005	14	5	14	50	41	3	25	14				

Fig. 5 : Répartitions des « *posts* » sur PearPC-devel et PearPC-users

Ainsi, d'un côté, la plupart des messages qui auraient leur place dans PearPC-users, c'est-à-dire les messages d'« utilisateurs finaux », avec tout ce que ce terme peut comporter de péjoratif, se retrouvent dans l'autre liste, celle des techniciens, celles des utilisateurs « *hard-core* ». Présenté face à ces deux listes, l'utilisateur final, le « *newbie* » (« *débutant* », au sens péjoratif du terme, parfois traduit en français par « *neuneu* »[33]), l'« imbécile », qui vient faire perdre leur précieux temps aux codeurs, aura le discernement nécessaire à comprendre que

32 Sur ce sujet, voir l'article de Nicolas Auray, *La régulation de la connaissance : arbitrage sur la taille et gestion aux frontières dans la communauté Debian,* 2003 : « *Parce que la visée de qualité impose une approche a priori laxiste en terme d'accueil, un problème structurel des communautés de connaissance consiste dès lors à bien régler leur degré d'ouverture : comment arbitrer entre l'effet club, qui bénéficie de la grande taille, et l'effet de congestion, qui incite à la clôture ? Quelle taille donner à la base des abonnés de la liste de discussion ? Comment gérer les hors sujets ? Faut-il scinder le forum en deux ?* »

33 On retrouve cette stigmatisation des « *newbies* » chez Auray, qui en propose une autre traduction, guère plus flatteuse : « [*A]insi, la stigmatisation des néophytes, surnommés « blaireaux » (newbies) lorsque dans les forums ils manifestent leur incompétence, rappelle la rudesse de l'accueil des nouveaux dans les saloons de la ruée vers l'or.* » (Auray, *L'Olympe de l'Internet français et sa conception de la loi civile*, 2002)

cette liste est en quelque sorte une voie de garage. Une liste créée par les gourous pour montrer que oui, le « *petit peuple* » les intéresse.

Autre explication à la désertion de cette seconde liste, les communautés parallèles qui se sont créées au fur et à mesure dans le but d'offrir aux débutants (toute notion péjorative a ici disparu) l'aide qu'ils cherchaient, entre non-techniciens. Ce sont des communautés d'utilisateurs pragmatiques qui cherchent à ce que le programme fonctionne, à ce que, en cinq minutes si possible, ils aient réussi à installer un programme qui, de toute façon, en l'état actuel des choses, ne leur sert à rien – car les performances et les limitations actuelles de PearPC ne permettent pas d'utiliser des logiciels Macintosh dans des conditions de travail acceptables.

Ces groupes de non-techniciens, tout intéressants qu'ils soient (on peut y voir, par exemple de très beaux outils didactiques s'y créer), ne seront que très peu abordés dans mon analyse. En fait, on peut les considérer comme une classe à part, comme les *utilisateurs*, des gens d'un autre monde qui s'intéressent à un projet sans que la tête de celui-ci ne s'en aperçoive – ou ne s'en préoccupe. Ils ne viendront troubler ces lignes que lorsque, excédés par le manque d'écoute des « gens d'en haut », une délégation de ces utilisateurs finaux (ou l'un d'entre eux, chevalier blanc et cavalier seul, se sentant appelé comme représentant syndical), viendra poser sa question sur la liste des développeurs. Qui rivaliseront de marques de mépris envers cet empêcheur de coder en rond.

Sur PearPC-devel, on discute **technique**. Ça discute technique, mais selon des règles très strictes, rappelées régulièrement par l'un des gourous de la liste – ou l'un de ses dauphins. Car, peut-être faut-il le préciser à ce stade, sur PearPC-devel, la hiérarchisation est forte. Nous sommes dans le domaine du *libre*, mais pas dans le domaine du *démocrate*. C'est une *méritocratie* exacerbée – fut-elle de bon aloi, elle n'en est pas moins stricte. C'est une méritocratie *tacite* : les rangs et les rôles sont clairement établis et tout le monde les connaît. Il sera très rare qu'un des réguliers sorte du rang et se permette de s'exprimer sur un sujet qui ne lui aurait pas été attribué, d'une manière paradoxalement aussi tacite qu'univoque.

III. Gravir les échelons : un guide pratique

Admettons que nous voulions accéder au sommet de l'échelle de PearPC-devel. Quelles sont les qualités requises, quels moyens mettre en oeuvre pour accéder à ce nirvana ?

La méthode *Sebastian Biallas (SB)*, jeune étudiant en informatique, quelque part dans la banlieue d'Aix-la-Chapelle, âgé de 23 ou 24 ans : pour lui, pas de problème : il est le père du projet. Lorsqu'en mai 2004 il publie son programme sur SourceForge.net, son aura de gourou éclate instantanément. Ne vient-il pas en effet de lâcher dans la nature un programme, certes imparfait, mais qui démontre que ce que l'on croyait jusqu'alors impossible était non seulement possible, mais aussi réalisé.

Et comme il l'a fait selon la méthode du logiciel libre, tout qui veut vérifier sa démonstration n'a qu'à télécharger les sources de son programme, qu'il a eu la bonne idée de délivrer sous la GPL, la *crème de la crème*, la formule magique qui assure à quiconque décide de s'y soumettre d'être suivi par une cohorte d'enthousiastes. La méthode est connue et assure à celui qui l'emploie, pour utiliser une expression que j'emprunte à John Lennon, un « *karma instantané* ».

Il n'est toutefois pas commode d'appliquer la méthode *SB*. En effet, réussir son effet de prestige – sortir de nulle part un programme prétendu impossible – lui a demandé un travail considérable et un don certain. La méthode n'est pas reproductible aisément, le talent ne pouvant apparaître par la seule volonté de l'acteur.

La méthode *Stefan Weyergraf (SW)* ne diffère guère de la précédente, puisque *SW*, ami intime de *SB*, a écrit des pans important du programme avec ce dernier et a été, de mai à juillet, son égal. Sa mort tragique, en juillet 2004, dans des circonstances qui font penser à un suicide, ne lui aura pas laissé le temps de savourer son aura de gourou – ou, peut-être, n'aura-t-il pas pu l'assumer.

Les circonstances de la mort de ce programmeur restent inconnues et n'ont pas été divulguées. La seule source disponible à ce sujet est le texte laconique suivant, publié sur la mailing-list PearPC-devel le 3 juillet 2004 :

> « Yesterday evening at 23:00 one of my best friends, one of my best critics, one of the most valuable programmers of PearPC, the one who could help me in all situations, the one with whom I had the best time of my life, died after getting hit by a train.
> Stefan 'steveman' Weyergraf -- Rest In Peace. We'll never forget you.

I'll continue PearPC. But I need some time to understand this dramatic accident. »

(« Hier soir à 23 heures, un de mes meilleurs amis, un de mes meilleurs critiques, un des programmeurs les plus précieux de PearPC, celui qui pouvait m'aider dans toutes les situations, celui avec qui j'ai passé les meilleurs moments de ma vie, est mort après avoir été heurté par un train.
Stefan 'steveman' Weyergraf – Repose En Paix. Nous ne t'oublierons jamais.
Je continuerai PearPC. Mais j'ai besoin d'un peu de temps pour comprendre cet accident dramatique. »

Cette paire de programmeurs, *SB* et *SW*, a représenté de mai à juillet, la plus haute autorité charismatique dans le microcosme PearPC-devel. Rigoureux au point d'en devenir rudes, presque grossiers avec ceux qui ne les suivaient pas dans leur méthodologie, ces « *dictateurs bénévoles*[34] » ont dès le départ instauré une discipline inflexible et fait comprendre à qui voulait joindre le projet que s'ils acceptaient les contributions de qualité avec enthousiasme, ils n'étaient pas prêts à laisser quiconque faire prendre à leur projet un autre cap que celui qu'ils s'étaient assignés, pendant les deux années où ils y avaient travaillé à deux, posant les fondations du logiciel dans l'anonymat.

Un exemple de cette rigueur fut donné par *SW*, dans le dernier e-mail qu'il posta à la liste, quelques heures avant son décès, où il voulait remettre les pendules à l'heure – et les lecteurs de PearPC-devel à leur place – après que plusieurs membres aient exprimé quelques critiques quant à la gestion du projet par *SB* et *SW*. Cet e-mail était intitulé « About this project » (« Au sujet de ce projet »), le 2 juillet 2004. En voici un extrait capital :

« But you should all understand that this is the first time we are trying to cooperate on a software project with more than 2 developers. Before we released everything to the public we were working as a team of two for years. Because of that Sebastian and I are a good team. Extending this cooperation to more than 2 developers is difficult. Suddenly we see ourselves confronted

34 « *Benevolent Dictator* » ou « *Benevolent Dictator for Life* », est le terme parfois choisi pour désigner, de manière humoristique, le ou les gourous responsables d'un logiciel, qui décide(nt) de la direction que prendra celui-ci. Ils ont généralement la confiance des utilisateurs du programme et des autres programmeurs de l'équipe, qui reconnaissent que le « dictateur » n'abusera pas de son pouvoir de manière arbitraire.

with a bunch of developers and with a community. This process takes up a lot of the concentration we had once spent in the actual work.

This frustrates us too. We don't want to spend too much work in coordination. Things should pretty much self-organize and work without much intervention. This doesn't mean that we all should not talk about things or that Sebastian and I don't want to answer questions but that we expect our co-developers to be independent. I'm not pointing at anyone here, saying that he/she/it did this or that, got on our nerves or anything.

We are furthermore expecting that you stick to our rules. An example are the coding rules. We will now apply patches that do not follow our coding style because we do not want patch work, not even on the level of aethetics. We can have and have had discussions about how this or that problem should be solved. And when in doubt we have chosen our own opinion. So yes there is dictate. And that is good because rules are needed.

We are expecting for *good programmers* to work with because we consider ourselves to be good (with the exception of being good team developers after all). And we're saying so without feeling any shame. We don't want to educate people, we want to work.
What makes up a good programmer? Does he have to know C++,C,Perl,Assembler or Haskell? No, bullshit. It's not about the language at all. It's not even very much about typing at all! It's about *thinking*. It's about *planning*. It's about *solving problems*. And still I'm not pointing at anyone saying that he/she/it isn't a good programmer.

We want to have quality code. We will not apply patches that introduce hacks. We have introduced hacks into the code ourselves (the PROM eg.) but we're trying to get rid of them very hard. Quality code is modular. Quality code doesn't repeat itself. No code duplication. But still there is tolerance for this.

If any of you is feeling that he can't work this way then we shall say goodbye and leave without too much anger right now. »

(« Mais vous devez tous comprendre que c'est la première fois que nous essayons de coopérer sur un projet logiciel avec plus que 2 développeurs. Avant d'avoir rendu public tout ce que nous avons fait, nous avons travaillé comme une équipe de deux pendant des années. Grâce à cela, Sebastian et moi formons une bonne équipe. Étendre cette coopération à plus de deux développeurs est difficile. D'un coup, nous nous voyons confrontés à plein de développeurs et à une communauté. Ce processus nous prend beaucoup de la concentration que nous utilisions avant pour faire le vrai travail.

Ça nous frustre également. Nous ne voulons pas passer trop de temps à la coordination. Les choses devraient plutôt s'auto-organiser et fonctionner sans notre intervention. Ça ne veut pas dire que nous ne parlerons pas de choses et d'autres ou que Sebastian et moi ne voulons pas répondre à des questions, mais nous attendons de nos co-développeurs qu'ils soient indépendants. Je ne montre personne du doigt, en disant qu'il/elle/ça a fait ceci ou cela, nous a mis a bout de nerfs ou quoi que ce soit.

De plus, nous attendons de vous que vous respectiez nos règles. Un exemple est les règles de codage. Nous n'appliquerons pas les patches qui ne suivent pas notre style de codage parce que nous ne voulons pas un patchwork, pas même au niveau de l'esthétique. Nous pouvons avoir et avons eu des discussions sur la manière dont tel ou tel problème devrait être résolu. Et quand nous étions dans le doute, nous avons choisi notre propre opinion. Donc, oui, il y a une dictature. Et c'est très bien, car des règles sont nécessaires.

Nous nous attendons à travailler avec des *bons programmeurs* parce que nous nous considérons nous-mêmes comme bons (avec l'exception de ne pas être de bons développeurs d'équipe après tout). Et nous disons cela ainsi sans ressentir aucune honte. Nous ne voulons pas éduquer des gens, nous voulons travailler.
Qu'est-ce qui fait un bon programmeur ? Doit-il connaître le

C++, le C, Perl, l'Assembleur ou Haskell? Non, conneries. Ce n'est pas une question de langage. Ce n'est même pas vraiment une question de taper ! C'est une question de *penser*. C'est une question de *planifier*. C'est une question de *résoudre des problèmes*. Et encore une fois, je ne montre personne du doigt en disant qu'il/elle/ça n'est pas un bon programmeur.

Nous voulons avoir du code de qualité. Nous n'appliquerons pas des patches qui introduisent des bricolages. Nous avons introduit des bricolages dans le code nous même (la PROM, par ex.) mais nous essayons très fort de nous en débarrasser. Le code de qualité ne se répète pas. Pas de duplication du code. Mais il y a quand même une tolérance pour cela.

Si n'importe qui d'entre vous a l'impression qu'il ne peut pas travailler de cette façon, nous nous dirons au revoir et il quittera sans trop de colère, tout de suite. »)

Ce texte, qui explique les raisons et motivations impliquées par une très grande rigueur, est également riche dans sa définition des valeurs fondamentales de PearPC-devel : **l'amour du travail bien fait** (que l'on retrouve également chez Himanen et Weber) et la **recherche d'excellence du code**, quitte à déforcer la valeur de l'éducation : le code ouvert permet de s'instruire par sa lecture, mais il n'est pas question de faire ses classes en soumettant du code de débutant à PearPC. Ce code serait rejeté.

La méthode *Daniel Foesch (DF)* est la première « vraie » méthode d'entrée dans le projet (les deux premiers étaient fondateurs du projet et donc en faisaient partie *de facto*) : programmeur très doué, il s'est investi dans le projet dès juin 2004, en développant un module qui permettait à PearPC de simuler des ordinateurs Macintosh plus récents que ceux émulés jusqu'alors[35]. Dès le début, il s'est montré rigoureux et a accepté les directives de programmations et la méthodologie imposée par la coalition au pouvoir, *SB* et *SW*. Ses e-mails étaient fréquents, longs, extrêmement bien documentés et, surtout, pertinents. Le travail fourni pour sa première contribution était impressionnant, même aux yeux de *SB* : « impressive work so far! » (*SB*, 6 juin 2004).

35 Techniquement, il a programmé l'émulateur du processeur de type G4. PearPC à l'origine émulait un processeur de type G3.

Il fut le premier à se voir attribuer des droits CVS sur le code source du programme : en simplifiant à l'extrême, on peut dire que lorsqu'un programmeur reçoit des droits sur le serveur CVS d'un programme, il a le droit d'en modifier le code sans avoir à demander au préalable la permission de le faire à l'un des programmeurs principaux. Ce faisant, il rejoint le noyau central de l'équipe, sans toutefois bénéficier des mêmes droits absolus que les fondateurs du projet.

Suite au décès de *SW*, *DF* a continué à travailler sur sa « branche G4 » avec assiduité et constance, jusqu'à être en mesure de s'autoproclamer, début 2005, « *PearPC's second lead developer* » (« *Second développeur à la tête de PearPC* »). Aujourd'hui, il a acquis une notoriété et une aura qui auraient sans doute été celles de *SW*, s'il avait vécu. Lors de l'incident que je relaterai dans quelques lignes, il a même commencé à émettre des opinions techniques qui divergeaient de celles du chef de projet, *SB*. Je reviendrai en détail sur ce point dans la section suivante.

La méthode *Rick Altherr (RA)* demande trois prérequis importants : participer ponctuellement au développement du programme en résolvant certains problèmes obscurs et a priori peu passionnants, le faire selon une méthode irréprochable et avoir un employeur prestigieux. Arrivé sur la mailing-list en octobre, ce n'est que le 15 novembre qu'il signe son premier e-mail de son adresse « *corporate* » (d'entreprise). Il a une adresse @apple.com, ce qui fait de lui un employé d'Apple. Il a donc accès à une quantité d'informations dont même *SB* ne pourrait rêver. Il prouve aussi, en s'investissant de la sorte dans le projet, que le département légal d'Apple a été prévenu de l'existence de PearPC et, non seulement n'a pas essayé de faire cesser le projet (ce qui serait légalement difficile), mais a également autorisé un employé de la firme à y contribuer.

Le premier mail de *RA* contenait ce texte énigmatique :

> « I managed to get pearpc running and to boot the Panther install CD on OS X using Apple's X11. (Yes, there is a reason for this madness, but I'm not allowed to say any more) » (25 octobre 2004)
> (« J'ai réussi à faire tourner PearPC et à démarrer le CD d'installation de Panther sur OS X, en utilisant le X11

d'Apple. (Oui, il y a une raison à cette folie, mais je ne suis pas autorisé à en dire plus) »)

Et son second mail contenait ceci :

« I know a fix for this, but I'm waiting on legal to determine if I can contribute it back. » (26 octobre 2004)
(« Je connais une solution pour ça, mais j'attends que le service légal détermine si je peux vous envoyer une contribution. »)

Enfin, le premier mail à l'occasion duquel il a fourni du code était également son premier mail signé de son adresse @apple.com.

Plus accessible sans doute, mais moins prestigieuse, **la méthode *Jens von der Heydt (JvdH)***. Dès mai 2004, ce membre s'est montré l'un des plus actifs de la liste, prenant part à toutes les discussions, répondant inlassablement aux nouveaux arrivants et à leurs questions, presque toujours identiques. Il a également écrit de nombreuses lignes de code pour le projet, mais toujours des parties maudites, populaires, plus proches des demandes des « *pragmatistes* » que de celles des « *architectes* ». C'est cependant toujours avec un grand soin qu'il a produit ces lignes de code, en prenant toutes les précautions d'usage pour ne pas froisser les têtes du projet.

Paradoxalement, il est devenu l'un des plus fidèles porte-parole des *architectes*. Il est celui qui comprend parfaitement les enjeux de ces derniers, mais n'a pas les connaissances techniques nécessaires à travailler aux parties les plus ardues du projet. C'est donc de manière très humble qu'il contribue aux parties annexes du projet, les « déchets » que ceux-ci ne voudraient approcher pour aucune raison[36]. Depuis un an, il s'affaire également à rechercher, par tous les moyens, de la documentation sur le système d'affichage vidéo des Macintoshes. S'il réussissait ce tour de force, il deviendrait lui-même l'un des « *architectes* » et obtiendrait probablement son accès CVS. Sa grande loyauté envers ses « supérieurs » fait de lui une person-

36 Ainsi, il a écrit pour PearPC les routines d'affichage SDL, un système qui permet à l'émulateur de gagner un peu en performances, très prisé par les utilisateurs finaux, mais ignoré – voire détesté – par les têtes du projet. Il a aussi travaillé à ce que PearPC continue à fonctionner sous la plate-forme Windows : les développeurs considérant Windows comme un jouet sans intérêt, ses contributions ont été acceptées, mais aucune de ses contributions n'aura aidé à mieux connaître le fonctionnement interne du Macintosh, diminuant leur valeur aux yeux des développeurs-en-chef.

nalité appréciée de la liste et le seul « non-*architecte* » à pouvoir participer à toute discussion sans avoir à craindre le mépris des « hautes instances ». Il représente la figure parfaite de ce que j'appellerai le *dauphin*.

IV. PearPC – quelques affaires

Karma, aura... Ces termes mystiques, assez répandus dans le milieu du libre, ne sont pas le fruit du hasard. Une des valeurs fondamentales auxquelles est attaché le programmeur libre est celle de la **reconnaissance, du respect**. Maximiser son aura, minimiser les possibilités d'être attaqué pour avoir commis en public le péché d'ignorance est le but commun de ces figures de proue de la communauté PearPC. En contribuant à PearPC-devel en faisant montre de l'intelligence la plus fine, le développeur se crée une véritable armure, un bouclier contre les attaques de groupes aux opinions divergentes. Et la force de cette armure – de cette aura – se mesure par l'apport des connaissances contribuées à la liste.

Car voici une autre valeur forte partagée par les membres de ces communautés : la **connaissance, le partage libre de l'information**. Comme le fait très justement remarquer Himanen, ils calquent leur méthodologie sur la méthodologie académique. Ils partagent, avec l'académie, avec l'université, un idéal de partage de leurs connaissances. L'information « veut être libre », ne peut être l'objet d'une marchandisation quelconque. Ce faisant, ils le clament haut et fort, leur production est *scientifique* – même si ce point fait l'objet de polémiques, polémiques dont la présentation dépasserait le cadre de ce texte.

Reconnaissance, connaissance, science. Voilà les trois mots-clefs qui vont nous permettre de comprendre comment est née une dispute sur la liste PearPC-devel, comment celle-ci a évolué, comment ses membres ont tiré profit ou « perdu des points » dans l'aventure et, finalement, comment s'est terminée cette dispute.

1. L'affaire « *lwarx* » et « *stwcx* »

Tout a commencé le 15 avril 2005, lorsque *DF* décide de poster un long mail où il explique, un peu hors sujet par rapport aux discussions alors en cours, comment il envisage une nouvelle méthode d'implémentation de deux instructions d'assembleur, « *lwarx* » et « *stwcx* » (l'assembleur est le langage de programmation réputé le plus compliqué, parce que le plus proche de la manière dont travaille l'ordinateur). À la lecture de cet e-mail, tout membre de la

liste se demande de quoi il peut bien parler, tant son contenu est compliqué, voire inaccessible et hors propos quant aux autres discussions parallèles menées lors de cette mi-avril.

Les réponses ne se font pas attendre : un habitué de la liste, habituellement d'un assez bon niveau, que nous n'avons pas encore présenté, *John Kelly* (*JK*), répond

> « Great message, but mentioning what the instructions were for at the top of the message would have helped. » (*JK*, le 15 mai 2005)
> (« Superbe message, mais mentionner à quoi servent ces instructions au-dessus du message aurait aidé. »)

En d'autres termes, il n'y a rien compris et souhaiterait savoir de quoi il retourne !

DF estime n'avoir de compte à rendre à personne :

> « I didn't feel a need to explain what they were, because anyone who as in the know, and would find the message useful would either already know the instructions, or would look them up. » (*DF*, le 15 mai 2005)

Le ton est sec : il ne lui semblait pas utile d'expliquer de quoi il retournait, parce que n'importe qui qui voudrait y comprendre quelque chose n'aurait qu'à déjà connaître le sujet, ou au minimum se documenter. Et de conclure son message par « I for one don't read any threads on subjects that I have no interest/knowledge in. » : lui ne perdrait pas son temps à lire des messages sur des sujets qu'il ne comprend pas.

Suivent quelques échanges, relativement incompréhensibles pour l'observateur que je suis, entre quelques « têtes pensantes » du projet, dont certaines ne s'étaient pas encore vraiment dévoilées jusqu'alors. À voir les réactions de *DF*, ces réponses sont « dignes » de son e-mail.

Que voyons-nous à ce stade du développement de l'incident ? Même s'il parle d'un problème de programmation, *DF* en fait, demande surtout une redistribution des cartes : son « problème technique », si je ne doute pas du fait qu'il souhaitait le soulever pour lui-même est cependant relativement anecdotique. Il veut résoudre ce problème technique, mais la

manière dont il exprime ce souhait est tellement ardue qu'il sait, en posant sa question, qu'il va susciter une vague d'incompréhension.

Lorsque *JK* pose sa question, il donne à *DF* une occasion de le ridiculiser en public, occasion que *DF* ne manque pas de saisir. En effet, depuis un certain temps, une certaine animosité entre ces deux développeurs se faisait sentir, mais, jusqu'alors, tout ce qu'on pouvait reprocher à *JK* était d'être attaché à la plate-forme Windows. Il n'aurait pas été *correct* de la part de *DF* de l'attaquer sur un choix personnel – la liberté de choix est un autre domaine sacré dans le milieu du développement libre. Il est par contre tout à fait *correct* de « dégrader » un programmeur pour son manque de **connaissances**[37].

Les quelques programmeurs qui auront répondu au courrier de *DF* en lui prouvant avoir compris quelque chose à son message, au contraire, auront gagné du galon. *Brian Onn* (*BO*), par exemple, aura eu la bonne idée de répondre au message de *DF* en y apportant des précisions et en répondant par l'affirmative à peu près toutes les remarques que *DF* avait pu faire quant à ces précisions. Les moins doués, quant à eux, ont gagné à se taire. Auraient-ils cherché à se montrer intéressants, *DF* les aurait couvert d'un mépris très étudié, à l'aide d'une des phrases assassines dont il a le secret.

Et *SB*, là-dedans ? Auteur de la première implémentation des instructions critiquées, il ne pouvait se permettre de rester silencieux très longtemps. Aussi a-t-il demandé très vite ce qu'il y avait de mauvais à son implémentation, pour finalement se voir répondre – entre les lignes et après quelques dizaines d'échanges de mails à haute teneur technique – qu'il n'y avait rien de vraiment mauvais dans son implémentation d'origine. Car mon intuition est que, en plus de résoudre un petit problème technique qui tenait à coeur à *DF*, cet incident avait surtout pour but de « faire le ménage », de changer la dynamique de la liste, de créer de nouvelles alliances, de séparer le bon grain de l'ivraie et, ce faisant, d'augmenter la qualité scientifique du logiciel et de ses développeurs, avec en arrière-plan le rejet des programmeurs « *Windows* » envers qui l'ire des têtes de liste n'a jamais été que très maladroitement dissimulée.

37 Tout comme il aurait été *correct* de l'attaquer pour son absence d'« **esprit scientifique** », ou, pour *DF*, de jouer sur son ancienneté et donc sur sa **reconnaissance acquise**.

2. La critique et la méthode scientifique comme modèles

Un caractère particulièrement présent dans la plupart des projets de logiciel libre est la grande foi, partagée presque unanimement par les développeurs, dans la méthode scientifique. C'est à mon sens le grand apport de l'ouvrage d'Himanen d'avoir mis en exergue cette foi en cette valeur très présente chez les développeurs de logiciel libre, et de l'avoir montrée comme l'une des raisons du succès du mouvement.

Ce n'est donc pas un hasard que *SB*, le créateur du projet PearPC, rappelle très régulièrement qu'il est important que les participants à la mailing-list soient **critiques**. Ainsi, dans un e-mail daté du 19 mai 2005, *Kyle Nesbit (KN)* écrit :

> « BTW, I do not mean to be critical, but I do not know how you are going to implement an SMP PPC emulator with out understanding PPC consistency model (...) I am very interested in seeing PearPC support SMP systems and am willing to contribute, but this is not going in the right direction. »
> (« À propos, je ne veux pas me montrer critique, mais je ne sais pas comment vous allez pouvoir implémenter un émulateur de PPC multiprocesseur (SMP) sans comprendre le modèle de cohérence du PPC (...) Je serais très intéressé de voir PPC supporter les systèmes multiprocesseurs et je désire contribuer, mais ceci ne va pas dans la bonne direction. »)

La réponse de *SB* ne se fait pas attendre :

> « Please be critical. It's an important aspect of science and development (...) I have to acknowledge that I have no idea of consitency models in general. »
> (« Sois critique. C'est un aspect important de la science et du développement (...) Je dois reconnaître que je n'ai aucune notion des modèles de cohérence en général. »).

Non seulement *SB* applique la méthode scientifique – et tout tout particulièrement l'aspect critique de celles-ci – , mais, avant de reconnaître humblement sa méconnaissance du sujet, il se fait lui-même épistémologue, rappelant que la critique est un aspect essentiel du

développement. Il est d'ailleurs remarquable que cette intervention « épistémologique » n'est pas un événement isolé : les commentaires de ce type sont légion dans ses écrits. Par exemple, presque un an plus tôt, le 24 juin 2004, en réponse à un *mea culpa* de *JK* où ce dernier s'excusait d'avoir déchargé sa mauvaise humeur en tenant des propos durs (« `I have to apologise for my last message. Things have been frustrating me today (...) Sorry for venting on you` », *JK*, 24 juin 2004), la réponse de *SB* avait été semblable :

> « `No problem. As long as you have good arguments, don't stop discussing.` » (*SB*, 24 juin 2004)
> (« `Pas de problème. Pour autant que tu aies de bons arguments, n'arrête pas cette discussion.` »)

SB demande d'ailleurs ouvertement à être critiqué :

> « `I see nothing bad in telling people that their code is bad. In fact I needed more people telling _me_ how bad _my_ code actually is.` ») (2 juillet 2004)
> (« `Je ne vois rien de mauvais dans le fait de dire à des gens que leur code est mauvais. En fait, j'aurais eu besoin que plus de monde _me_ dise combien _mon propre_ code est mauvais.` »)

Même en des moments autrement plus difficiles, comme le 3 juillet 2004, dans son mail-épitaphe pour son meilleur ami et second développeur de PearPC, *SW*, il reconnaît à ce dernier la grande qualité d'avoir été « `one of my best critics` » (« `l'un de mes meilleurs critiques` »).

Corollaire de ce choix méthodologique, la plupart de ses e-mails sont écrits sur un ton froid, détaché et laconique : souvent, c'est en quelques mots (littéralement) qu'il apporte une réponse à de longues questions qui lui sont posées. Parfois, c'est un simple oui, ou un simple non, qui répond à un développement de plusieurs dizaines de lignes de texte. Technique qui n'est pas sans rappeler la maïeutique socratique, qu'il emploie à dessein, probablement pour gagner du temps – il répond généralement à ses e-mails « par lots » –, mais surtout pour encourager ses lecteurs à réfléchir à un sujet, avant de le développer plus avant.

3. Une gestion du temps chaotique

Si Himanen se montre assez enthousiaste quant à la thématique du « *temps flexible* », (Himanen, 2001, p. 29 et ss.) il est important de noter que sa notion est assez idéalisée. La principale objection que je pourrais faire à cette conception est qu'il décrit un monde où le *hacker* dispose librement de son temps et organise son temps de travail en fonction de ses envies, qu'il attribue son temps libre au codage et que, par exemple, il préférera travailler de nuit que selon un horaire classique, de 9 à 18 heures.

Ce faisant, Himanen redistribue le travail le long de la semaine, mais semble penser que la régularité sur un plus long terme (le mois ou l'année) du *hacker* est identique à celle du calviniste weberien et que, de semaine en semaine, une quantité de travail comparable est abattue.

Nous allons voir que, dans le cas des développeurs PearPC, la distribution du temps est beaucoup plus chaotique. Il peut arriver, par exemple, que deux semaines s'écoulent sans que pratiquement aucun message ne soit posté sur la liste, tandis que les deux semaines suivantes seront particulièrement fécondes.

Voici comment *DF* annonça, le 15 novembre 2004, non sans un certain humour, un « *horrible obstacle au développement* », premier d'une longue série :

> « I've hit a nearly impenetrable roadblock in development at this time. It is called, World of Warcraft Open Beta.
> If you're aware of it, then you understand my complete inability to code at this time, as all my free time is being sucked up by this truely marvelous and great game.
> I'll eventually dull my play time down so that I can continue developing PearPC, but as for now, I'm quite busy playing WoW. »
>
> (« Je viens de me heurter à un obstacle de développement à peu près incontournable pour l'instant. Ça s'appelle World of Warcraft Open Beta.
> Si vous êtes au courant de son existence, vous comprendrez alors mon incapacité complète à coder pour l'instant, comme

tout mon temps libre est aspiré par ce grand jeu vraiment merveilleux.
Je prévois de diminuer mon temps de jeu à l'avenir, de manière à pouvoir continuer à développer PearPC, mais, pour le moment, je suis assez occupé à jouer à WoW. »

Voici que nous voyons entrer en scène un nouvel ennemi du programmeur *FLOSS* et non des moindres : le jeu vidéo. Cet e-mail fut suivi de messages compatissants, bien entendu toujours à prendre au second degré : *JvdH* :

« Lol, I know what you're talking about. Now and then a brilliant game comes out and all my spare time and work time suffer from it :) » (*JvdH*, 15 novembre 2004)
(« *Lol*, je comprends de quoi tu parles. Parfois, un superbe jeu sort et tout mon temps libre ainsi que tout mon temps de travail en souffrent :) »)

JK seconde ça : « I'll third that ». Car en effet, si ce problème ne peut être généralisé (*SB*, par exemple, semble immunisé contre le phénomène), on peut considérer qu'une grande partie des développeurs de la liste « souffre » de cette passion-soeur de la passion de l'informatique, qui meut leur participation au projet.

Au-delà de l'aspect anecdotique de cet échange d'e-mails, on peut dresser un constat plus général : si une passion vient se substituer à leur passion originelle, cette nouvelle passion aura tendance à phagocyter la majeure partie du temps libre de la communauté. S'il s'agit d'une passion individuelle (par exemple, tel membre décide d'apprendre à jouer de la guitare), le projet n'en souffre pas vraiment, le problème se confinant à un seul des développeurs. Il en va tout autrement lorsque, comme dans l'anecdote développée ci-dessus, une passion est collective et découle d'un phénomène de mode : non seulement constaterons-nous une diminution de l'activité de la liste, mais aussi ses membres pourront-ils auto-renforcer leur passion, comme le montre ce courrier, suite du même fil de discussion, posté par *DF* :

« For anyone who cares, I'm in the retail, and here's my vitals:
Name: Rotemilch

Server: Doomhammer
Guild: (will be) Dracones Antiquitatis, or Dracones Veritatis » (« Si ça intéresse quelqu'un, je suis sur le marché et voici mes données (d'identification dans le jeu) (...) ») (*DF*, 27 novembre 2004)

Très vite, JK répond qu'il est également inscrit sur ce jeu et communique ses coordonnées : « I'm on the Eonar server (EST) as Marintok and am a Tauren Hunter. » (*JK*, 27 novembre 2004)

Heureusement pour la communauté, ce type de mode est relativement éphémère et, moins d'un mois plus tard, les développeurs retournent généralement à leurs premières amours. Remarquons également, sur un note plus positive, que la passion commune pour les jeux vidéos d'un nombre important de contributeurs de PearPC-devel renforce leur sentiment d'appartenance à une même communauté.

Autre embûche, beaucoup plus régulière – et planifiée – est qu'un certain nombre de contributeurs au logiciel est encore étudiant. Ce sont alors les périodes d'examens qui marquent de grandes périodes de silence. Chez *SB,* par exemple, qui s'est fait discret sur la liste dès le 1er mai 2005, et n'a carrément rien posté entre le 18 juin et le 21 juillet 2005. Il a été tellement occupé, d'ailleurs, pendant cette période, que c'est *DF* qui s'était chargé d'expliquer son absence de la liste, le 3 juillet 2005 :

« Sebastian has been quite busy for a long time with finishing school, so he's been unable to really work a lot on PearPC. » (« Sebastian est très occupé depuis longtemps à terminer l'école et a donc été incapable de travailler vraiment beaucoup sur PearPC. »)

Quelques mois plus tôt, d'ailleurs, le même *DF* avait répondu à des critiques comme quoi le projet n'avançait plus très vite, par ces mots :

« Yes, our main developers are pretty swamped. We also have to earn a living by working at a job, or going to school. Thus, our time to work on PearPC is limited. If someone were to

guarentee me $20,000 a year payed out in half-month increments, I'd spend 40 hours a week or more on PearPC. Until then, I need a job to pay off my student loans. ($400 a month sucks when it's not getting a good job yet)
(...)
Real life takes precedence over PearPC unfortunately, I must eat, and I must pay my student loan. That's just the way the world works. I don't believe Sebastian has earned enough from donations to really pay rent for a single month, let alone afford to distribute any cash to other developers, so they can work full time on PearPC. » (*DF*, le 15 mars 2005)

(« Oui, nos développeurs principaux sont plutôt débordés. Nous devons aussi gagner notre vie en travaillant à un boulot, ou aller à l'école. Dès lors, notre temps passé à travailler sur PearPC est limité. Si quelqu'un venait à me garantir 20 000 dollars l'an, payés par tranches de deux semaines, je passerais 40 heures par semaine ou plus sur PearPC. D'ici-là, j'ai besoin d'un travail pour payer mon prêt d'étudiant. (400 dollars par mois, c'est pénible, quand on n'a pas encore un bon boulot)
(...)
La vraie vie a malheureusement priorité sur PearPC, je dois manger, et je dois payer mon prêt d'étudiant. C'est tout simplement comme ça que le monde fonctionne. Je ne crois pas que Sebastian ait gagné assez en donations pour payer son loyer pour un seul mois, et encore moins pour distribuer de l'argent aux autres développeurs, pour leur permettre de développer PearPC à plein temps. »)

C'est en cela que je ne rejoins pas Himanen : bien souvent, dans les communautés de programmeurs *FLOSS*, la « vraie vie » est une cause majeure d'irrégularité. Le *hacker*, chez Himanen, est d'office soustrait aux exigences bassement matérielles, n'a pas de factures à payer, de bouches à nourrir[38]. De même, chez Himanen, sa passion pour la programmation prédomine sur toute autre passion ou manière d'employer son temps libre. Himanen n'exclut pas, bien entendu, que le *hacker* ait d'autres passions, ou du moins d'autres manières de gérer

38 Auray émet une réserve semblable : « *Certes, les conditions objectives d'existence d'un tel être académique ne sont jamais analysées sont renvoyées en un revers de main (sic) – la condition de possibilité du hacker est ainsi d'avoir réussi à régler les problèmes de survie, alors même que c'est leur analyse qui fait toute la richesse du travail de Bourdieu.* » (Auray, *De l'éthique à la politique : l'institution d'une cité libre*, 2002)

son temps[39], mais, ce faisant, il n'est jamais décrit comme étant en proie à un conflit de passions (les jeux vidéos, par exemple) ou à un conflit entre ses passions et sa « vraie vie ».

On pourrait, bien sûr, attribuer cette distortion, cette vision idéale, au fait que le modèle du *hacker*, chez Himanen, en l'occurrence Linus Torvalds, a su faire de sa passion sa profession. Or, si aujourd'hui, il est de plus en plus fréquent que ces programmeurs soient effectivement employés par des entreprises pour développer du logiciel libre, la vision idéalisée d'Himanen n'en reste pas moins une exception : un grand nombre de logiciels libres est développé selon un modèle proche de celui qui nous préoccupe dans ce chapitre, PearPC, par des hobbyistes ou des étudiants et ne peuvent donc être confondus avec l'occupation professionnelle principale de leurs programmeurs.

4. L'attachement aux valeurs de la GPL

Le 12 octobre 2004, la nouvelle de la parution d'un programme concurrent à PearPC se répand sur les sites technologiques comme une traînée de poudre. MXS, Inc., une start-up Hawaïenne, annonce avoir développé un émulateur Macintosh pour le système d'exploitation Windows. Parmi les caractéristiques de nouveau logiciel, baptisé CherryOS, on peut lire qu'il est extrêmement rapide : selon les dires de son développeur, CherryOS permet d'exécuter Mac OS X, le système d'exploitation d'Apple, ainsi que ses applications, à 80% de la vitesse du processeur hôte. Rappelons que PearPC ne permet qu'une vitesse 20 fois moindre à celle du processeur hôte – ou, exprimé en pourcentage... 5% !

La promesse d'une telle vitesse, théoriquement impossible à atteindre par un procédé d'émulation, suscite bien entendu très vite la curiosité des lecteurs de tous les sites technologiques ayant relayé la rumeur, Slashdot en tête. La curiosité, mais aussi la méfiance, qu'une telle affirmation ne peut que susciter chez les spécialistes de l'informatique, qui commencent très vite à chercher des preuves de ce qu'annonce Arben Kryeziu, le programmeur qui aurait écrit CherryOS suite à la frustration qu'avait engendré chez lui le fait de devoir transporter deux ordinateurs portable, un PC et un Macintosh lors de ses

39 « *A hacker may join his friends in the middle of the day for a long lunch or go out with them for a beer in the evening, then resume work late in the afternoon or the next day. Sometimes he or she may spontaneously decide to take the whole day off to do something completely different.* » (« *Un hacker peut rejoindre ses amis au milieu de la journée pour un long déjeuner ou aller boire une bière avec eux en soirée, puis recommencer le travail tard dans l'après-midi ou le jour suivant. Parfois, il ou elle peut décider spontanément d'employer une journée complète pour faire quelque chose de complètement différent.* » (Himanen, *The Hacker Ethic*, 2001, p. 33)

déplacements. Pour seulement 49,99 dollars, MXS Inc. proposait donc son logiciel, en en gardant la source fermée.

Bien entendu, la nouvelle de l'existence de ce « concurrent » se propage très vite sur PearPC-devel. C'est *Eric Harmon* (*EH*), un « *lurker* », peu habitué à poster sur la mailing-list, qui est le premier à évoquer le sujet, le 13 octobre, dans un courrier intitulé « `CherryOS - You knew someone would mention it` » (« `CherryOS - Il fallait bien que quelqu'un le mentionne` ») :

> « `So what is everyone's take on this? Many people think perhaps they have ripped off PearPC. Info @ Slashdot:` http://apple.slashdot.org/apple/04/10/12/1622247.shtml?tid=179&tid=190&tid=201&tid=3
>
> `Thoughts?` »
>
> (« `Alors, quelle est la position de tout le monde à ce sujet ? Beaucoup de monde pense qu'ils ont peut-être vampirisé PearPC. Infos sur Slashdot.`
>
> `Votre avis ?` »)

À ce stade, la prudence quant à ladite vampirisation s'imposait : en effet, aucune version de démonstration de CherryOS n'était disponible au téléchargement (mais MXS Inc. promettait de pallier à cette lacune dans les plus brefs délais). Toutefois, des vidéos de démonstration permettaient de voir, fût-ce sommairement, à quoi ressemblait le produit.

Fig. 6 : `CherryOS, extrait de la vidéo de démonstration`

Très vite, le fil de discussion « CherryOS - You knew someone would mention it » devint l'un des fils les plus commentés de l'histoire de PearPC-devel. C'est un autre « *lurker* », *Lucas Menge* (*LM*), qui fit la première observation intéressante (ou, peut-être, la recopia d'une discussion lue sur un quelconque site de technique informatique) :

> « you know what's weird?
> look at this (...) anyone find the white bar between the titlebar
> and the actual screen a little familiar? :\ » (*LM*, 13 octobre 2004)
> (« Vous savez ce qui est bizarre ?
> regardez ceci (...) [Fig. 6, NDA] quelqu'un trouve-t-il la barre blanche entre la barre de titre et l'écran [émulé] vaguement familière ? :\ »)

À ce moment, tout était dit : pour tout qui avait suivi le développement de PearPC depuis le début, cette barre blanche était effectivement très familière : historiquement, elle avait été utilisée dans les toutes premières versions de PearPC pour afficher un bouton permettant de changer le CD virtuel de la machine émulée, bouton entre-temps abandonné ; personne n'avait jugé nécessaire d'en enlever la barre blanche qui servait de support à ce bouton, probablement dans le but de le réhabiliter dans le futur. D'autres similitudes furent rapidement trouvées, comme le choix de la même touche – F11 – pour « libérer » la souris de l'émulation et quelques autres détails (la manière de désigner le CD-ROM émulé, le texte de démarrage « now starting client... », certaines limitations arbitraires de la taille du disque dur émulé par PearPC).

La première réaction de *SB* fut très concise, comme à son habitude, puisqu'il se contenta de « linker » une capture semblable à la Fig. 6, la commentant d'un « Hmmmmm.... :) ». *JvdH* alla plus loin, puisqu'il mesura « scientifiquement » la taille de ladite barre blanche :

> « I checked the size of the "change-cd" - bar with photoshop on that JPG. And guess what, it's 28 pixels height. and that does ring a bell :) pearpcs gMenuHeight is 28.
> but we'll see.... » (*JvdH,* 13 octobre 2004)

Celle-ci avait une hauteur de 28 pixels[40]. Or, comme le remarqua *JvdH*, c'est précisément la taille assignée à la bande blanche dans le code source de PearPC, ce qui faisait passer la probabilité d'une pure coïncidence de « très improbable » à « quasi impossible ».

En tout, ce sont 42 messages distincts relatifs à CherryOS qui furent postés ce seul 13 octobre 2004 (un pic !), chacun apportant une petite précision ou interrogation quant à ce mystérieux logiciel, venu de nulle part et – le jour même où la liste avait été informée de son existence – un nombre conséquent de preuves, qui devenaient irréfutables si on les additionnait, avait été trouvées, qui montraient que CherryOS comportait, au minimum, de grandes parties du code de PearPC. Et donc, bafouait les principes de la GPL et était une imposture.

Imposture. C'est d'ailleurs le titre du premier mail posté le 14 octobre par *Dean Beeler* (*DB*) : « `I have strong reason to believe CherryOS is a fraud` ». *DB*, qui avait écrit certaines parties du code permettant l'émulation préliminaire du son dans PearPC, avait fait son « `rather fascinating detective work` » (« `travail de détective plutôt fascinant` ») et avait trouvé que le créateur de CherryOS, par le passé, avait déjà violé la GPL et s'en était déjà défendu à coups de mensonges honteux. Un autre « *lurker* », habitant Hawaï, alla même jusqu'à se déplacer jusqu'aux locaux de MXS, Inc., pour vérifier l'existence de cette société-fantôme.

Ce « travail de détective » réalisé par la communauté sur l'affaire CherryOS dura encore plusieurs mois[41]. Il se mêla à d'interminables logorrhées sur la GPL, mélangeant citations du texte et interprétations personnelles, où l'on put voir, par exemple, *DF* conseiller de lire la GPL dans le texte, le 20 octobre :

> « `If you have any further questions on this matter, I'd suggest you consult the GPL. If you find any direct proof (cited paragraph, and sentence of the GPL) that disproves my argument and validates yours, please let me know.` »
> (« `Si tu as d'autres questions sur le sujet, je te suggérerais`

40 Un *pixel* (PICture ELement) est la plus petite unité de mesure de l'affichage d'un moniteur. Elle correspond à un point carré. Un écran d'ordinateur typique, à l'heure où j'écris ces lignes, affiche plus ou moins 1024 sur 768 pixels.

41 Il me reste à préciser, en toute humilité, que j'avais moi-même, titillé par cet événement, participé à – et créé – l'un des fils de discussion, « `CherryOS - the conspiracy theory` » (« `CherryOS - la théorie de la conspiration` » ; l'observation participative a du bon !).

de consulter la GPL. Si tu y trouves une preuve directe (un paragraphe cité, et une phrase de la GPL) qui contredit mon argument et valide le tien, merci de me le faire savoir. »)

Lorsque la version de démonstration de CherryOS fut finalement disponible, il apparut immédiatement que ce logiciel était effectivement une version de développement de PearPC, augmentée d'une interface graphique de piètre qualité, écrite par la société MXS, Inc. Son « programmeur » n'avait même pas pris la peine, probablement par incompétence, de déguiser le code de PearPC, qu'il était alors facile de comparer, via un éditeur hexadécimal ou un désassemblage sommaire, avec le vrai logiciel.

Cette certitude établie, *DF* et *SB,* propriétaires à eux deux des droits sur une grande partie du code de PearPC, ainsi que quelques autres membres de la communauté, envisagèrent sérieusement d'attaquer MXS Inc. en justice, et en appelèrent à la *Free Software Foundation* afin de défendre leur propriété intellectuelle.

MXS Inc. fut également victime de pressions de la part de nombreux défenseurs indignés du logiciel libre et de la GPL, mais c'est la presse informatique en ligne qui joua un rôle décisif de médiateur dans cette affaire. La société hawaïenne, ou plutôt Arben Kryeziu, en personne – il ne fut jamais établi clairement si la MXS Inc. avait d'autres employés que ce dernier – commença à tenter de justifier, par l'intermédiaire de différentes interviews accordées à ces « webzines »[42], à l'aide d'un discours maladroit et révisé à chaque fois qu'une incohérence y était détectée par une communauté de plus en plus irritée.

Il argumenta, dans un premier temps, qu'il n'avait pas utilisé de code de PearPC, que les mêmes idées avaient logiquement les mêmes conséquences et que ce simple fait expliquait les similitudes entre les deux programmes. Ce qui n'expliquait pas, par exemple, la présence dans le programme de CherryOS de la chaîne de caractères « SPIRO MULTIMAX 3000 », un terme dénué de sens, inventé par *SB* pour désigner une variable à laquelle il ne trouvait pas de meilleur nom ! Kryeziu révisa donc son histoire et annonça qu'un ancien employé de MXS Inc. avait « emprunté » certaines parties de PearPC, mais que cet employé avait été congédié et qu'une nouvelle version de CherryOS serait bientôt disponible.

42 Voir par exemple, cet article du célèbre magazine en ligne *Wired* : http://wired.com/news/mac/0,2125,65368,00.html

La nouvelle version, une fois disponible, s'avéra être PearPC, à peine mieux camouflé. L'ire de la communauté augmenta d'un cran et finalement, Kryeziu annonça que CherryOS serait dorénavant disponible « selon une licence Open Source ». Ce qui n'arriva jamais : CherryOS et le site qui lui servait de support disparurent dans les premiers mois de 2005. L'affaire était terminée.

Que nous montrent ces réactions rapides, ces enquêtes spontanées, ces centaines de lignes de glose sur cette tentative désespérée d'une petite start-up de se faire connaître ? Nous avons vu combien cette contravention à la GPL aura littéralement **mobilisé** la communauté libre, combien celle-ci pouvait se montrer **attachée aux valeurs** représentées par cette licence. Nous avons vu que sa **force de mobilisation**, pour sauvegarder le principe du logiciel libre, était très forte et que celui qui oserait ignorer ces valeurs pouvait s'attendre à un retour de flamme conséquent. Nous avons également vu le **rôle d'arbitre que prend alors la presse informatique en ligne**, qui se sera révélé capital dans la lutte des développeurs pour la défense de leur code.

L'autre volet intéressant de cette affaire est la **connaissance pointue de la GPL**, de ses tenants et aboutissants, de son historique et, surtout, de son esprit, que manifestent les membres de la communauté PearPC-devel. C'est alors qu'on peut **mesurer pleinement l'attachement à un texte, tant à sa lettre qu'à son esprit**, qui anime nombre d'entre eux. Ainsi, *DF*, l'espace de quelques jours, mit de côté son rôle de développeur (« *PearPC's second lead developer* »), pour se muer en ardent défenseur de ce texte souvent qualifié de « sacré ».

5. Rattrapé par la réalité

Au cours de ces quelques dernières sections relatives à PearPC, le dédain affiché pour le système d'exploitation Windows – et pour son créateur, Microsoft – est apparu à plusieurs reprises. Nous l'avons vu dans les premiers chapitres de ce travail, Microsoft est la société qui symbolise le mieux le logiciel fermé et le refus des valeurs du libres. Microsoft est également très souvent stigmatisé pour leur production de logiciels de mauvaise qualité.

SB et *DF* font assurément partie de ces détracteurs, ils l'ont assez souvent répété. Ils programment sous Linux, adhèrent aux valeurs de la GPL, à une certaine conception de la qualité du logiciel.

« but then I disagree with the whole MS domination thing anyways, so, I may be a little biased. » (*DF*, 29 avril 2005)
« mais aussi, je ne suis pas d'accord avec toute cette histoire de la domination de MS [Microsoft], de toute façon, donc, je suis peut-être un peu biaisé. »

C'est donc avec un peu d'étonnement que je pu lire sur PearPC-devel le message suivant, signé par *DF* le 28 juillet 2005 :

« As some of you may know, and others maybe not. I've taken a job with Microsoft.
I'm legally encumbured right now from working on the PearPC code.
If someone can contact me regarding assistance with the development of the AltiVec code, it'd be greatly appreciated. »
(« Comme certains d'entre vous peuvent le savoir, et d'autres pas. J'ai accepté un boulot chez Microsoft.
Je suis légalement empêché dans l'immédiat de travailler sur le code de PearPC.
Si quelqu'un peut me contacter au sujet d'une assistance avec le développement du code AltiVec, ce serait grandement apprécié. »)

On aurait pu s'attendre à quelques boutades quant au choix de l'employeur de *DF*. Il n'en fut rien. Au contraire, ce furent des réactions enthousiastes et des congratulations que reçut *DF* :

« Congrats, Daniel! I'm happy to hear that you finally found something. » (*JvdH*, 28 juillet)
« Yes, congratulations Daniel. » (*BO*, 28 juillet)

Il est cependant assez clair que *DF*, tant qu'il sera employé par Microsoft, aura d'énormes difficultés à continuer à travailler sur PearPC et même sur tout code GPL, dont nous avions pourtant vu qu'il était un ferveur défenseur :

« Except when you're working for Microsoft, who has a big long list of do's and don'ts with respect to open source. Literally, the only "this is ok to do without permission" is to download FOSS software for private internal use, and use ONLY. They're pretty clear in their available FOSS legal materials (which is one of the first thing that I've familiarized myself with here) that they just can't... er... don't trust FOSS at all. They're concerned obviously about leaks of IP (which is what Microsoft is all about) » (*DF*, 28 juillet 2005)

(« Sauf que quand tu travailles pour Microsoft, qui a une grande longue liste de « fais » et « ne fais pas » par rapport à l'open source. Littéralement, le seul « c'est OK de le faire sans permission » est de télécharger du logiciel FOSS pour son usage privé interne et de l'utiliser SEULEMENT.
Ils sont assez clairs dans leurs documents légaux disponibles sur le FOSS (qui est une des premières choses avec lesquelles je me suis familiarisé ici) qu'ils ne peuvent simplement pas... euh... avoir confiance du tout en le FOSS. Ils sont évidemment inquiets quant aux fuites de propriété intellectuelle (qui est la raison d'être même de Microsoft) »)

Cette dernière affaire est intéressante en ce sens qu'elle **contraste avec la vision utopiste d'Himanen sur les rapports des *hackers* à l'argent et au travail**. Suite à une proposition d'emploi bien rémunérée – ces détails n'ont pas été discutés sur la mailing-list, mais quelques fouilles dans l'historique de *DF* sur Slashdot, par exemple, informent qu'il a accepté de déménager, lui et sa famille, dans un autre état (*DF* vit en Amérique), suite à une proposition juteuse de Microsoft –, *DF*, un apologiste de la GPL « au-dessus de tout soupçon », a été rattrapé par l'éthique protestante weberienne et sa vie a finalement pris le tournant d'un chemin « capitaliste ».

Il ne s'agit bien entendu pas d'accomplir une quelconque généralisation ni justement de valeur. Bien des programmeurs n'accepteraient de travailler pour une société *closed source* à aucune condition. Mais ce type de revirement n'est pas rare dans la communauté – les grosses entreprises « chassent les cerveaux » également dans ce vivier – et je trouvais que cette dernière affaire, en guise d'épilogue a cette analyse de PearPC-devel, avait la qualité d'aider à relativiser l'*éthique hacker*. Et pose la question de la mesure de l'intransigeance, de la force de l'attachement aux valeurs libres dont font montre les programmeurs libres.

V. PearPC – Conclusion

L'analyse approfondie des quinze mois de l'existence de cette mailing-list, sous l'éclairage théorique des deux premières parties de ce travail, nous montre une communauté libre *in vivo* et nous permet de nous immiscer dans son quotidien, son intimité. Sous la loupe grossissante de cette analyse, certaines notions que nous avions étudiées sont exacerbées, d'autres sont ignorées, voire reniées. Ce « petit » projet, cela va de soi, ne renseigne pas précisément sur le fonctionnement de plus gros projets – ni même d'autres petits projets parallèles, de taille similaire. Tout au plus nous fournit-il quelques indices.

Du côté des similitudes avec notre théorie, nous retrouvons l'**attachement à la méthode scientifique**, la **volonté d'accumuler de la connaissance, du savoir**. Une **forte hiérarchisation**, une **forte décentralisation** (les deux développeurs principaux, *SB* et *DF*, vivent respectivement en Allemagne et au Nouveau-Mexique et ne se sont jamais rencontrés). Un **amour du travail bien fait** (les contributeurs qui veulent rejoindre le projet doivent être « bons », le code doit être élégant) et des **personnalités charismatiques** à la tête de la communauté se comportant en « **dictateurs bénévoles** », garants de la qualité du projet. L'**attachement à un texte sacré et sa connaissance** – la GPL en l'occurrence – sont également très présents, tout comme la force de mobilisation que ces valeurs sont capables de produire.

Cette théorie insistait moins sur un fait pourtant prévisible : si les développeurs de PearPC sont heureux de montrer leur code au monde entier, **il n'est pas question que des développeurs débutants ou moins « bons » contribuent au code**. PearPC n'est pas une école de programmation, *SW* le dit clairement dans son dernier e-mail. La volonté de partage de connaissance est bien là, mais nous voyons se profiler un sens unique, qu'un informaticien définirait comme une connaissance « en lecture seule » et non en « lecture – écriture ». Cette conséquence est finalement le **corollaire de l'amour du travail bien fait**.

Par rapport à la théorie d'Himanen – je l'ai déjà indiqué à plusieurs reprises – je n'ai pas retrouvé un « **temps flexible** » ni une **éthique de l'argent** aussi marqués, aussi univoques, une même rupture avec le modèle weberien que chez le philosophe finlandais. Probablement parce qu'Himanen tente – et échoue sur ce point – de voir en l'*éthique hacker* un nouveau modèle politique et économique, là où je ne vois tout au plus qu'un nouveau sous-système s'inscrivant parfaitement dans le capitalisme.

Conclusion

I. Quelques réponses

La réalité recouverte par le logiciel libre et *Open Source* est grande et, malgré le grand soin apporté à sa présentation et à son analyse, je n'ai pu aborder que les quelques facettes du phénomène que je trouvais les plus pertinentes quant à ma problématique de départ.

J'ai néanmoins recueilli, grâce à cette analyse, des réponses aux questions que je posais dans l'introduction de ce document. Réponses qui ne prétendent en rien à l'exhaustivité, ni à l'infaillibilité.

Au fil de ces paragraphes, nous avons appris que ces communautés sont constituées majoritairement – ce n'est pas une surprise – de programmeurs informatiques, mais que différents enthousiastes et entrepreneurs, sans compétences informatiques particulières, peuvent joindre ces communautés en leur fournissant une aide, chacun à sa manière (un artiste fournira des éléments d'interface graphique, un juriste des conseils juridiques, un pédagogue – ou quiconque manie habilement sa langue – proposera des modes d'emploi).

La communauté *FLOSS* est relativement fragmentée, mais se rejoint sur un corpus restreint de valeurs. Les valeurs qui font l'unanimité sont le **partage de l'information**, l'**ouverture**, le **don**, l'**attention à l'égard des autres**, l'**amour du travail bien fait**. Une certaine foi en la valeur de l'**éducation** est également souvent présente.

Certaines valeurs sont réservées aux « factions » les plus extrêmes du mouvement. La notion de **liberté**, au sens stallmanien du terme, n'est pas partagée par tous. Certains préfèrent une définition de la liberté plus lâche, plus proche du concept de **liberté individuelle** (Raymond), que de celui d'une **liberté communautaire** (Stallman).

Contrairement à un cliché répandu, **personne – ou très peu de monde –, au sein de ces communautés, ne remet en cause le capitalisme**. Seule la recherche de profit réalisée en **dissimulant l'information** est condamnée. Certains condamnent ce comportement de

manière univoque et entière (Stallman), d'autres, au contraire, admettent des concessions et des alliances avec des entreprises vendeuses d'information, quand des intérêts commerciaux qu'ils considèrent supérieurs sont en jeu. Pas plus que le « cliché communiste », le « cliché du voeu de pauvreté » n'est vrai : refuser de dissimuler l'information n'est **pas vraiment un problème dans un système économique privilégiant la notion de service sur celle de produit**. Le libre et l'ouvert sont des notions nées dans un monde capitaliste et ont évolué au sein de ce système. Les membres des communautés *FLOSS* pensent généralement faire bon ménage avec le capitalisme.

Ces communautés se définissent positivement, « pour » les valeurs que je viens d'évoquer, mais aussi négativement, **« contre » certaines conceptions de l'informatique, de la politique, des affaires, contre certaines lois**. Ainsi, les **brevets logiciels**, la censure, les programmes informatiques médiocres et le marketing volontairement trompeur de certaines sociétés informatiques sont-ils généralement vus comme autant d'ennemis. Souvent, cet ennemi est représenté – caricaturé ? – par une seule et même entreprise, considérée comme la mère de tous les maux : Microsoft.

À la tête de ces communautés se trouvent différents **personnages charismatiques**, les « gourous ». On les trouve à différents niveaux : certains se veulent les porte-parole de toute une communauté (Stallman, Raymond) et écrivent des textes à portée philosophique, voire d'« *anthropologie amateur*[43] ». D'autres sont les « **despotes éclairés** » d'une subdivision de ces groupes : la taille de ces subdivisions varie énormément, selon les projets : ainsi, Linus Torvalds est à la tête d'un groupe de dizaine de milliers d'adeptes, tandis que nous avons été présentés aux « gourous » de communautés beaucoup plus petites (*SB*, *SW* et *DF* dans PearPC-devel).

Certains de ces « gourous » se montrent très **paternalistes** face à leurs membres. Les passions égalitaires ne sont – peut-être paradoxalement – pas très présentes au sein de ces communautés. Un certain élitisme règne dans beaucoup d'entre elles. Devenir membre d'une de ces communautés demande, dans un premier temps, des qualités d'abnégation. Il est ensuite souvent nécessaire de faire ses preuves pour y être accepté. **Les membres passifs sont tolérés**, à condition qu'ils ne prennent pas la parole en public au nom d'une communauté dont ils n'ont pas pleinement intégré les valeurs.

---

43 Eric Raymond, cité par Auray (*Le savoir en réseaux et l'empreinte inventive*, 2000)

Une des caractéristiques de ces communautés est leur forte **décentralisation**. Cependant, **le niveau de hiérarchisation en leur sein est variable selon les communautés**. Parfois, une forte hiérarchisation peut concentrer en un même espace physique un grand nombre de programmeurs, ce qui amène à ces projets un semblant de centralisation. Les communautés de programmeurs libres **tolèrent mal la bureaucratisation** et utilisent la technologie de manière à se libérer de celle-ci.

Souvent, les membres de ces communautés ne se sont pas rencontrés et seuls leurs avatars électroniques communiquent. **L'émergence**, au cours des années 90, **du réseau Internet** est l'**une des raisons**, mais aussi l'**une des conditions d'existence** de ces communautés. Celles-ci ont adopté toutes les technologies qu'Internet pouvait leur offrir pour permettre une collaboration efficace : e-mail, mailing-lists, forums, sites web, serveurs de fichiers.

Le mouvement dispose de textes sacrés. La **connaissance de la lettre et de l'esprit** de ces derniers par nombre de ses membres et **la propension de ceux-ci à en discuter les détails les plus intimes** (je pense par exemple à la connaissance de la GPL et aux longues discussions que celle-ci occasionne) est très forte. L'attachement des membres au texte sacré de leur choix a un caractère qui évoque parfois un dogmatisme de type religieux.

Tous ces éléments, ajoutés à l'âge respectable de ces communautés (22 ans si on compte la création de la *Free Software Foundation* comme acte fondateur du mouvement) semblent indiquer que **nous avons à faire à un véritable mouvement plus qu'à un phénomène de mode**. Un mouvement qui, s'il arrive à dépasser ses querelles internes et à passer le cap de son « adolescence », est là pour durer.

II. Perspectives

L'informatique est un domaine où les générations se succèdent à un rythme effréné. La loi de Moore[44] me semble appropriée pour déterminer la taille du cycle d'une génération informatique : dix-huit mois. Un monde si changeant incite à la prudence.

44 Gordon Moore, un des co-fondateurs de la société Intel, qui fabrique aujourd'hui les microprocesseurs qui équipent la grande majorité des ordinateurs personnels, a donné son nom à une loi empirique, qu'il a formulée en 1965, et qui veut que la complexité des circuits intégrés (et donc des microprocesseurs) doublerait tous les dix-huit mois. Depuis lors, même si, d'année en année la « fin de la loi de Moore » est annoncée, cette prédiction s'est révélée remarquablement juste.

Ceci dit, je ne résisterai pas à la tentation de clore ce travail par quelques mots sur l'avenir possible du logiciel libre, qui me fut inspiré en grande partie par une des intuitions d'Himanen. Selon ce dernier, la raison du succès du logiciel libre (« Académie ») est qu'il constitue un moyen plus adapté à la création de l'information que le modèle fermé (« Monastère »). Si cette hypothèse se vérifie, ce n'est plus vraiment une extrapolation que d'affirmer que le rêve de Stallman pourrait se réaliser d'ici une ou deux générations... informatiques.

Cette idée n'est pas sans faire appel à un certain darwinisme sauvage, motivé par le fait que beaucoup de logiciels commerciaux ont aujourd'hui atteint une maturité certaine. Prenons un exemple : l'évolution d'un des logiciels les plus connus – probablement *le* plus connu –, Microsoft Word. Il existe bien quelques différences entre sa version « Word 95 », sortie il y a dix ans et sa version actuelle, mais force est de reconnaître que ces différences sont mineures – et même invisibles aux yeux du profane. L'explication en est simple : malgré tous les efforts de recherche et développement que Microsoft injecte à son traitement de texte, ce produit a atteint sa maturité. La même chose pourrait être dite des autres applications de la suite bureautique Microsoft Office, dont la version basique regroupe Word (traitement de texte), Excel (tableur) et PowerPoint (logiciel de présentation par « slides »).

Du côté de la concurrence libre de Microsoft Office, nous trouvons, entre autres logiciels, OpenOffice.org. Si celui-ci pouvait encore sembler maladroit, difficile à utiliser, « *buggé* » il y a encore un an ou deux, ses versions actuelles approchent à grand pas de la maturité – pour l'anecdote, ce mémoire a d'ailleurs été entièrement rédigé sous OpenOffice.org. Parce que Microsoft Office a, lui, atteint cette maturité et, en admettant que le modèle du logiciel libre soit, comme le suppose Himanen, meilleur, on peut poser l'hypothèse que, dans peu de temps, les avantages liés à l'utilisation de la suite Microsoft Office seront réduits à néant.

Ses inconvénients, par contre, qui rebutent tant Richard Stallman, ont tendance à augmenter au fur et à mesure que sa liberté diminue : son coût est bien entendu le plus évident, mais également la politique de *lock-in* de Microsoft, qui incite à constamment renouveler les licences, rend difficile la coopération entre entreprises qui n'utilisent pas la même version de Microsoft Office, ou les ambitions de la firme d'encrypter les documents générés par ses logiciels, afin de lier irrémédiablement ces documents aux solutions Microsoft (les fameux DRM – « *Digital Rights Management* »).

Selon ce raisonnement, seules des campagnes de marketing de plus en plus agressives pourraient continuer à maintenir la suprématie de Microsoft Office face à ses son concurrent libre (et sous GPL) OpenOffice.org. Et il est douteux que cette prolongation en vie artificielle dure longtemps.

La même démonstration pourrait être appliquée à d'autres logiciels propriétaires, publiés aussi bien par Microsoft que par d'autres.

J'apporterai une petite nuance à cette affirmation. Je pense, en effet, qu'un type de logiciel, très répandu, est fondamentalement incompatible avec ce raisonnement et avec le logiciel libre en général – du moins tel que nous le concevons en 2005. Il s'agit des logiciels de jeu, dont l'industrie se rapproche de plus en plus de celle de la musique ou, plus encore, de celle du cinéma. Pour cette raison – en plus de différentes raisons techniques qu'il serait difficile d'expliquer en ces quelques lignes –, les jeux « libres » sont aujourd'hui très rares et ceux qui existent ne sont pas « compétitifs » par rapport aux jeux commerciaux.

Allons encore un peu plus loin. Si le marché de l'informatique personnelle devait en venir entièrement au libre, peut-être pourrions-nous assister, à la prochaine génération informatique, à une division plus ou moins complète de l'outil, selon le type de développement de ses logiciels : l'ordinateur personnel, de plus en plus ouvert – ainsi que sa déclinaison portable – accueillerait le logiciel libre, tandis qu'un autre acteur, qui vit parallèlement avec l'ordinateur personnel depuis plus de vingt ans, la console de jeu, deviendrait l'hôte de prédilection du logiciel fermé et propriétaire. Le modèle de développement forgerait l'outil !

Bizarre ? Pas sûr : les trois géants du jeu, Sony, Nintendo et Microsoft promettent, pour l'année 2006 (ou 2007), une « troisième » génération de consoles de jeu – respectivement la Sony Playstation 3, la Nintendo Revolution et la Microsoft XBox 360, qui semblent attirer les regards de tous les fanatiques du jeu vidéo. Que Microsoft – dont le Windows est aujourd'hui pour ainsi dire la seule plate-forme (hors consoles) accueillant les jeux – se retrouve dans cette liste pourrait être un signe qu'ils ont, eux aussi, senti le vent tourner.

Enfin, en plus des ordinateurs personnels et des jeux vidéo émerge une troisième catégorie : le *media center*, qui promet quant à lui de remplacer très bientôt le magnétoscope, le lecteur DVD et la chaîne hi-fi. Il est aujourd'hui très difficile de dire si ces machines seront équipées de logiciel libres – la première machine de ce type à avoir rencontré un succès

commercial, aux États-Unis du moins, la *TiVo*, tourne sous Linux – ou si leur degré d'intégration les rapprochera (les fusionnera ?) avec les consoles de jeu. Rendez-vous dans quelques années.

Bibliographie

Ouvrages

- *The Hacker Ethic* (2001), Pekka HIMANEN, Random House
- *La négociation des valeurs* (1998), Olgierd KUTY, De Boeck Université
- *Petites leçons de sociologie des sciences* (1993), Bruno LATOUR, Éditions de la Découverte
- *Du bon usage de la piraterie, Cultures libres, sciences ouvertes* (2004), Florent LATRIVE, Exils Éditeur
- *Les formes de l'échange : contrôle social et modèle de subjectivation* (2004), Claude MACQUET, Didier VRANCKEN, Éditions de l'Université de Liège
- *The Cathedral and the Bazaar* (2001), Eric RAYMOND, O'Reilly
- *Just for Fun: The Story of an Accidental Revolutionary* (2001), Linus TORVALDS, David DIAMOND, Harper Business
- *L'éthique protestante et l'esprit du capitalisme* (2002), Max WEBER, Les classiques des sciences sociales, Université du Québec

Articles, travaux et conférences

- Séminaire « *Le virtuel et le tangible : ce qui résiste* » (2005), département Costech de l'Université de Compiègne (France)

- *Le prophétisme hacker et son contenu politique* (1998), Nicolas AURAY, in revue Alice, n°1, octobre

- *Le savoir en réseaux et l'empreinte inventive* (2000), Nicolas AURAY, in revue Alice n°3, février 2000

- *La place des hackers dans l'innovation informatique : une comparaison des cas hollandais, français et américain* (2001), Nicolas AURAY, in E-Usages

- *De l'éthique à la politique : l'institution d'une cité libre* (2002), Nicolas AURAY, in Multitudes, n°8

- *La Licence Publique Générale GNU [logiciel libre]* (1999), mémoire, Mélanie CLÉMENT-FONTAINE, Faculté de Droit de Montpellier

- *Ethos technicien et information. Simondon reconfiguré par les hackers* (2002), Nicolas AURAY, in Gilbert Simondon. Une pensée opérative, J. Roux (éd.), Presses Universitaires de Saint Etienne

- *L'Olympe de l'Internet français et sa conception de la loi civile* (2002), Nicolas AURAY, in Cahiers du Numérique, vol. 3, n°2, Hermès

- *La régulation de la connaissance : arbitrage sur la taille et gestion aux frontières dans la communauté Debian* (2003), Nicolas AURAY, in Revue d'économie politique, pp. 74-79

- *Sociologie d'un annuaire de sites internet* (2004), Christophe LEJEUNE, Université de Liège (ULg), Faculté d'économie, de gestion et de sciences sociales (E.G.S.S.)

- Conférence « *La vigilance chez les administrateurs de sites Internet* » (2005), Christophe LEJEUNE, in séminaire « Le virtuel et le tangible : ce qui résiste »

- *The Cathedral and the Bazaar* (1997), Eric RAYMOND, in http://www.tuxedo.org

- *Homesteading the Noosphere* (1998), Eric RAYMOND, http://www.catb.org/~esr/writings/homesteading/homesteading/

- *The Magic Cauldron*, Eric RAYMOND (1999), http://www.catb.org/~esr/writings/magic-cauldron/

- *The GNU Manifesto* (1985), Richard STALLMAN, http://www.gnu.org/gnu/manifesto.html

- *Why Software Should Not Have Owners* (1994), Richard STALLMAN, http://www.gnu.org/philosophy/why-free.html

- *About the GNU Project* (1998), Richard STALLMAN, http://www.gnu.org/gnu/thegnuproject.html

- *The Pragmatist of Free Software: Linus Torvalds Interview* (1997), Hiroo YAMAGATA, http://hotwired.goo.ne.jp/matrix/9709/5_linus.html

- *Jargon file* (1975 – 2005), Collectif, http://www.catb.org/~esr/jargon/

Articles de presse

- *Le brevet divise les grands et les petits*, Mathieu VAN OVERSTRAETEN, la Libre Belgique, le 5 juillet 2005

- *Les deux camps crient victoire*, Anonyme, La Libre Belgique, le 6 juillet 2005

- *Fin de course pour la directive « brevet des logiciels »*, Véronique LEBLANC, la Libre Belgique, le 7 juillet 2005

Autres

- *GNU General Public License* (1991), Richard STALLMAN & Free Software Foundation, Inc., http://www.gnu.org/copyleft/gpl.html

Table des matières

Remerciements **4**

Introduction et méthodologie **5**

I. Choix de l'objet **5**

II. Le défi d'un terrain virtuel **6**

III. La contrainte linguistique **7**

IV. La problématique en quelques questions **8**

Partie I : Le logiciel libre **10**

I. Quelques considérations préliminaires **11**

II. Présentation des licences **12**

1. Copyleft : la GPL **13**

2. Non-copyleft : la licence BSD **17**

3. Comparaison entre les licences libres **20**

III. Quelques définitions **21**

1. Slashdot **22**

2. *Hacker* **23**

3. Mailing-list **23**

IV. Brève introduction historique **23**

V. Gourous **24**

1. Richard Stallman **24**

1. Idéologie **24**

2. Naissance du projet *GNU* et de la *Free Software Foundation* **27**

3. Une contribution sociale **29**

2. Linus Torvalds **30**

1. Linux, précisions terminologiques **30**

2. Linux, position dans le projet GNU **31**
3. Linus le gourou ? **32**
3. Eric Raymond **34**
1. « *The Cathedral and the Bazaar* » (1997) **35**
2. L'« *Open Source Initiative* » (1998) **36**
Partie II : Références théoriques **37**
I. Pekka Himanen **38**
II. Max Weber – quelques notions issues de la théorie de « *L'éthique protestante et l'esprit du capitalisme* » **39**
III. Mise en rapport de l'éthique protestante et de l'éthique *hacker* **41**
1. L'académie et le monastère **41**
2. L'éthique du travail **45**
3. Rapport du *hacker* au temps : « le temps flexible » **46**
4. L'éthique de l'argent **48**
5. Reconnaissance par la communauté **49**
6. Les valeurs protestantes *vs* les valeurs *hackers*, chez Himanen **50**
7. Conclusions sur l'oeuvre d'Himanen **50**
Partie III : Analyse empirique **52**
I. Des pommes, des poires et des... gnous **53**
II. La hiérarchie de la Poire **56**
III. Gravir les échelons : un guide pratique **57**
IV. PearPC – quelques affaires **65**
1. L'affaire « *lwarx* » et « *stwcx* » **65**
2. La critique et la méthode scientifique comme modèles **68**
3. Une gestion du temps chaotique **70**
4. L'attachement aux valeurs de la GPL **74**
5. Rattrapé par la réalité **79**
V. PearPC – Conclusion **82**
Conclusion **83**
I. Quelques réponses **84**
II. Perspectives **86**
Bibliographie **90**
Table des matières **94**