UNE ANALYSE SOCIO-POLITIQUE DU MOUVEMENT DU LOGICIEL LIBRE

Antoine Mazières

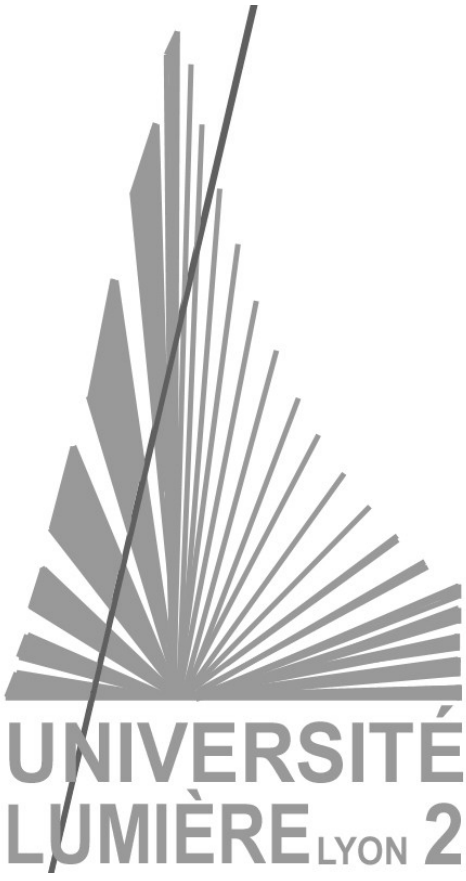
| mazieres{at}lavabit{dot}com |

Sous la direction de **(Prof. Dr. Tom Dwyer)**

Mémoire présenté à la Faculté de Droit et Science Politique (FDSP) de l'Université Lumière Lyon II.

Mémoire présenté le 13 août 2009 dans le cadre du programme de recherche du département de Science Politique (DCP) de l'Institut de Philosophie et Sciences Humaines (IFCH) de l'Université d'Etat de Campinas (UNICAMP, Brésil) comme pré-requis à l'obtention du titre de Maître en Science Politique.

UNICAMP



FICHE CATALOGUE RÉALISÉE PAR LA BIBLIOTHÈQUE DE L'IFCH (UNICAMP)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IFCH – UNICAMP
Bibliotecária: Maria Silvia Holloway – CRB 2289

M457u

Mazières, Antoine Bernard Marie
Uma análise do movimento do software livre e de código aberto / Antoine Bernard Marie Mazières. - - Campinas, SP : [s. n.], 2009.

Orientador: Thomas Patrick Dwyer.
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Filosofia e Ciências Humanas.

1. Software livre. 2. Hacker. 3. Programação (Computadores) – Aspectos políticos. 4. Programação (Computadores) - Pragmática. I. Dwyer, Thomas Patrick. II. Universidade Estadual de Campinas. Instituto de Filosofia e Ciências Humanas. III.Título.

Título em inglês: A sociopolitical analysis of the free software movement.

Palavras chaves em inglês (keywords) :
Free software
Hacker
Computer programming management – Political aspects
Computer programming management – Pragmatics

Área de Concentração: Ciência Política

Titulação: Mestre em Ciência política

Banca examinadora: **Thomas Patrick Dwyer, Maria Conceição da Costa, Valeriano Mendes Ferreira Costa**

Data da defesa: 13-08-2009

Programa de Pós-Graduação: Ciência Política

Résumé

Ce mémoire cherche à présenter les significations politiques et culturelles d'un Mouvement du Logiciel Libre considéré comme un ensemble hétérogène de communautés et de projets. De plus, à partir d'un historique de l'objet « Logiciel » depuis son origine, nous montrons qu'il a été différencié du matériel (*Hardware*) et, ainsi, fermé comme un produit fini par les entreprises de softwares naissantes. Dans ce contexte, le Mouvement du Logiciel Libre se présente autant comme une réaction au phénomène de « blackboxing », que comme une continuation de la tradition de libre-échange d'informations au sein de l'ingénierie informatique. Pour cela, il se structure à travers plusieurs aspects de l'éthique Hacker et de son agnosticisme politique pour construire une alternative technologique concrète. Ainsi, nous pouvons affirmer que les caractéristiques socio-politiques des communautés du logiciel libre doivent être recherchées dans l'acte même de la programmation, dans sa pragmatique, en tant qu'art ou régulation. De cette manière, nous étudions les cas spécifiques de plusieurs communautés (gNewSense, Samba, BSD) pour tenter de systématiser leurs positionnements techniques et socio-politiques envers le mouvement technologique contemporain.

Mots-clés Logiciel Libre ;
Hacker ;
Programmation (ordinateur) – Aspects politiques ;
Programmation (ordinateur) – Pragmatique.

Abstract

This dissertation presents some political and cultural significations of a Free Software Movement, understood as a heterogeneous aggregation of projects and communities. Then, the historical analysis of the "software object" shows how it become, in the first place, differentiated from the hardware and, then, secondly, closed as an end-product by the rising software companies. In this context, the Free Software Movement presents itself as a reaction to *black-boxing* phenomena, as well as a continuation of the computater engineering tradition of sharing knowledge freely. Therefore, FS Movement has become structured through diverse blends of Hacker Ethic and its own political agnosticism, in order to build a concrete technological alternative. This leads to the argument that sociopolitical characteristics of Free Software communities should be found in the very act of programming, and in its pragmatics as an art or a regulation. Finally, specific cases of several communities (gNewSense, Samba, BSD) are examined in an attempt to systematize their sociopolitical and technological positions of the contemporary technological movement.

Tags Free Software;
Hacker;
Computer programming management – Political aspects;
Computer programming management – Pragmatics.

Resumo

Esta dissertação procura apresentar as significações políticas e culturais de um movimento de Software Livre e de Código Aberto (SL/CA) entendido como conjunto muito heterogêneo de comunidades e projetos. Ademais, a partir de um histórico do objeto "software" desde a sua origem, mostramos como ele foi diferenciado do hardware e depois encerrado como um objeto fechado pela companhias de software nascentes. Nesse contexto, o movimento SL/CA aparece tanto uma reação ao fenômeno de blackboxing, como uma continuação da tradição de compartilhamento de informações dentro da engenharia da computação. Por isso, estrutura-se ao redor de vários ramos da ética hacker e de seu agnosticismo político para constituir uma alternativa tecnológica concreta. Isto nos permite afirmar que as características sociopolíticas das comunidades do Software Livre devem ser procuradas no próprio ato de programar, na pragmática, como arte ou regulação. Dessa forma, estudamos os casos específicos de varias comunidades (gNewSense, Samba, BSD) para tentar sistematizar os seus posicionamentos tecnológicos e sociopolíticos a respeito do movimento tecnológico contemporâneo.

Palavras-Chaves Software Livre;
Hacker;
Programação (Computadores) – Aspectos Políticos;
Programação (Computadores) – Pragmáticas.

Table des matières

Listes de Sigles

ADN	Acide DésoxyriboNucléique
API	Application Programming Interface
APL	Apache Public License
BBC	British Broadcasting Corporation
BIND	Berkeley Internet Name Domain
BSD	Berkeley Software Distribution
CEGE	Comitê Executivo do Governo Eletrônico
DMCA	Digital Millenium Copyright Act
DNS	Domain Name Server
DOS	Disk Operating System
F/LOSS	Free/Libre Open Source Software
FOSS	Free and Open Source Software
FSF	Free Software Fundation
FTP	File Transfer Protocol
gNS	gNewSense
GNU	GNU is Not Unix
GPL	General Public License
GPV	General Public Virus
GSM	Global System for Mobile communications
HTML	Hypertext Markup Language
IDABC	Interoperable Delivery of European eGovernment Services
IOSN	International Open Source Network
IP	Internet Protocol
iPDP	iPhone Development Program
IRC	Internet Relay Chat
JPEG	Joint Photographic Experts Groups
LGPL	Lesser General Public License
MIT	Massachussetts Institute of Tecnhology
MPEG	Moving Picture Experts Group
MPL	Mozilla Public License
MS	MicroSoft

NPL	Netscape Public License
ONU	Organisation des Nations Unies
OOo	OpenOffice.org
OSOR	Open Source Observatory and Repository
OSS	Open Source Software
PC	Personal Computer
PDA	Personal Digital Assistants
PGP	Pretty Good Privacy
SDK	Software Development Kit
SLCA	Software Livre e de Código Aberto
SMTP	Simple Mail Transfer Protocol
SO (*OS*)	Système operationnel (Operational System)
SOFTEX	Associação para Promoção da Excelência do Software Brasileiro
SSH	Secure SHell
SSL	Secure Sockets Layer
TMRC	Tech Model RailRoad
UNDP	United Nations Development Programs
VoIP	Voix sur IP
XML	eXtensible Markup Language

Table des figures

$? | ?

Le meilleur moyen de s'entrainer [pour devenir un bon programmeur] c'est d'écrire des programmes, d'étudier d'excellents programmes que d'autre personnes ont écrits...**vous devez vouloir lire le code des autres personnes** et ensuite écrire le votre, puis appeler les autres pour revoir votre code.

Bill Gates, 1989

Regarder le code comme seulement "des instructions pour la machine" c'est appauvrir une perspective comme regarder Shakespeare uniquement comme " de l'encre sur du papier".

Samir Chopra et Scott Dexter, **2007**

Introduction

Le mouvement du logiciel libre est une réaction à la privatisation du code qui s'est opéré dans les années 1960. Depuis lors il s'est structuré en différentes branches complexes et est parvenu à proposer une alternative technologique concrète dans divers domaines de la société de l'information, principalement dans l'utilisation et la construction de l'Internet.

De cette manière, on peut souligner plusieurs changements qui mettent en lumière le succès du logiciel libre, ou, du moins, son actualité dans les politiques publiques, les débats politiques et la littérature universitaire. En premier lieu, depuis plus d'une dizaine d'années, le nombre de dispositifs informatiques qui accompagnent le quotidien des individus des classes moyennes a crû de manière inédite. Le téléphone portable et l'ordinateur personnel sont deux objets communicacionnels dont les frontières s'estompent avec la réduction de la consommation d'énergie et de taille, tout en développant de meilleures performances et une stabilité accrue. Les moyens ainsi offerts deviennent des outils élémentaires de nombreuses activités qui, jusqu'alors, ne profitaient pas des effets de réseaux. Dans ce contexte, le logiciel libre apparait dans le sens commun comme un défi technique permettant la participation des utilisateurs au-delà de ce que permettent les plate-formes propriétaires.

De plus, parmi les idées qui circulent à propos du logiciel libre, on trouve celle d'une communauté sans frontières, dont l'accès est défini par la simple participation de ceux qui la souhaite. L'image de l'utilisateur-développeur se rapproche alors de celle d'un citoyen qui participe à un mouvement collectif : le mouvement technologique. Le "*You*" (Toi/Vous), élu personne de l'année en 2006 par le journal TIME, représente bien ce courant contemporain qui, d'une part, place l'individu comme actant décisif de la dynamique sociale, et, d'autre part, le fond entièrement au sein de la communauté à laquelle il participe. Dans ce sens, le mouvement du logiciel libre est un exemple qui donne beacoup de sens à cette "modernité", montrant des réalisations comme l'Internet, des possibilités, comme la particiaption politique directe et des problématiques, comme celle du modèle économique de l'appropriation

des savoirs.

Enfin, dans n'importe quelle direction qu'aille le mouvement technologique actuel, il est influencé par ses actants et une majorité de la population mondiale n'a pas accès aux réseaux et outils technologiques fondamentaux. C'est dans les pays périphériques que la fracture digitale est la plus apparente. Les classes moyennes ont accès aux technologies de l'information alors que les classes populaires accumulent un retard dans leur éducation digitale et, ainsi, ni ne participent, ni ne profitent de ces évolutions, bien que celles-ci paraissent structurer profondemment les sociétés contemporaines. Dans ce contexte, le logiciel libre se présente comme une solution adaptée aux politiques publiques d'inclusion digitale, étant donné qu'il permet d'offrir des technologies à coûts réduits et avec un potentiel d'adaptabilité majeur.

Dans cette étude, l'analyse se concentre sur les aspects culturels du mouvement et principalement sur ses déterminants politiques. Cependant, les communautés "libres" se présentent sans discours ou agendas politiques précis, développant ainsi une culture indépendante, comme branche de la culture "hacker". Néanmoins, bien que cet "agnosticisme politique" soit caractéristique du mouvement du libre, il est, dans le contexte informationnel, le visage d'une pratique sociale intense, informelle et normative qui construit l'aura politique du mouvement, entendu comme un ensemble. Ainsi, une hypothèse explorée dans cette recherche est que la pratique de la programmation, entendue comme pragmatique, est le vecteur constitutif et déterminant des pratiques politiques informelles et de ses conséquences sociales. De cette manière, nous présentons comment cette étude est systématisée.

Dans le premier chapitre, nous présentons une brêve histoire de l'émergence, de la constitution et de la consolidation du mouvement du logiciel libre dans le monde. En revenant sur les origines de l'objet Logiciel dans l'histoire des débuts de l'informatique, il se démontre que les distinctions "naturelles" faites entre le logiciel (*software*) et le matériel (*hardware*) pouvaient être interprétées différemment avant que des compagnies, comme AT&T ou Microsoft, viennent isoler un objet "logiciel" et fermer son code-source pour disponibiliser un produit fini avec des options de configuration et d'appropriation limitées. Dans ce contexte, le mouvement du logiciel libre apparaît comme une réaction, c'est à dire, un effort de maintenir un tradition de libre interaction dans le processus d'innovation, et de construire une alternative au modèle propriétaire qui se fortifie avec la production à grande échelle des ordinateurs personnels. En son sein, le mouvement a des structures communautaires, juridiques et institutionnelles complexes qui rendent difficile la délimitation d'une ligne d'action commune, au-delà de vouloir promouvoir la nécessité d'un code-source ouvert pour les logiciels. Néanmoins, la division institutionnelle entre logiciel "libre" et "ouvert" qui se manifeste en 1998,

permet de mieux comprendre les logiques économiques et idéologiques des diverses branches du mouvement global, et, ainsi, mettre en lumière ses défis contemporains.

Plus loin, le second chapitre tente de reconstruire une partie de l'effort théorique fait par la littérature universitaire traitant des aspects culturels du mouvement du logiciel libre et de la culture Hacker. Ainsi, la notion d'éthique hacker, entendue comme construction sociale, permet d'isoler plusieurs caractéristiques des actants du mouvement du libre. Cependant, comme l'affirment les travaux de l'anthropologue Gabriella Coleman [Coleman and Golub, 2008], le concept d'éthique hacker reste encore prisonnier du binaire moral mediatique qui s'est formé autour de l'image du hacker (Pirate ou Héros de l'aire digitale). De cette manière, il y a une tentative d'isoler le discours politique apparent comme un agnosticisme politique, dont la fonction est de permettre à la passion pour l'information qu'ont les hackers, de s'exprimer librement. Cette pratique structure ses politiques informelles, technologiques et pratiques que nous tenterons de réduire à une suite d'idéaux-types.

Toujours dans le second chapitre, nous présentons le paradigme théorique des pragmatiques de la programmation, construit pour interagir avec la recherche technique et bibliographique. Nous soulignons alors que l'acte d'écrire du code se réalise dans un contexte de normes et d'expectatives qui déterminent son esthétique. Ceci se réalise en comparant le fait de programmer avec celui de réaliser une oeuvre d'art, une loi ou une architecture physique, à partir de références universitaires. Ainsi, nous présentons le code comme un discours normatif et politique sur l'information et son traitement par la machine et le réseau. De manière générale, l'idée est de montrer que la pragmatique de la programmation est le vecteur de l'"agnosticisme politique" du mouvement, du fait d'être le vecteur de ses politiques informelles, et que les logiques du code qui écartent les références politiques traditionnelles sont celles qui articulent le discours normatif du mouvement sur l'information.

Dans le troisième et dernier chapitre, nous proposons une typologie des trois pragmatiques emblématiques de divers projets *open-source*, qui aura pour but d'interagir avec les typologies des politiques informelles du mouvement du logiciel libre. Ainsi, les logiques de transgression, de civisme technologique, d'inversion et de collaboration sont réinterprétées par le vecteur des pragmatiques de liberté, d'ouverture et de sécurité. Par l'analyse des études de cas, des références techniques et des concepts théoriques qui se trouvent dans l'effort technologique du mouvement, il s'établit des génériques d'interprétations des contextes politico-technologiques par les propres actants et communautés.

Chapitre 1

Historique et Définition

Selon Paul Ceruzzi, il y a deux courants d'histoire qui se sont construits autour de l'objet logiciel [Ceruzzi, 1998]. Le premier de ces courants, qui évite une description trop technique pour être destinée à un large public, s'intéresse à la formation des compagnies privées individuelles (IBM, Apple, Microsoft). Le seconde historiographie, elle, se concentre sur la naissance des langages de programmation (FORTRAN, COBOL, entre autres), en questionnant comment chaque langage est apparu et a permis à ses développeurs d'extraire de nouvelles fonctionnalités du matériel.

Depuis les travaux de Ceruzzi, un troisième courant d'historiographie s'est construit petit à petit autour de l'observation du mouvement du logiciel libre durant les années 1980 et 1990. Cette nouvelle narrative du mouvement technologique informatique rend compte des traditions de la culture hacker de programmation. Depuis ses origines dans les laboratoires universitaires d'informatique dans les années 1950 et 1960, s'est contruit l'histoire de comment des facteurs économiques et idéologiques ont formé les traditions culturelles de la programmation et comment ces récentes traditions ont affecté et même déterminé, quel type de logiciel est en train d'être écrit.

Ce chapitre introductif a pour objectif d'utiliser librement ces trois courants d'historiographie de l'informatique pour présenter le mouvement du logiciel libre dans son contexte historique, social et économique. Dans un premier temps, nous montrons comment le concept de logiciel s'est formé comme une abstraction du matériel et nous décrivons les éléments concrets que recouvrent, aujourd'hui, ces deux notions (1.1). Ensuite, nous expliquons comment l'objet logiciel s'est construit à grande échelle, en accompagnant l'histoire de l'ordinateur personnel et des compagnies qui l'ont fabriqué. Ainsi, on peut comprendre les premières traçes de la culture *open-source* comme une réaction à la privatisation du code (1.2). De plus, dans la troisième partie, nous montrerons la constitution des communautés et licences qui forment le

mouvement, ainsi que les institutions qui le promeuvent. De cette manière, à travers les conflits d'interprétation sur ce qu'est être "libre" ou "ouvert" par rapport à une technologie, on observe l'hétérogénéité du mouvement qui appelle à postuler d'une définition systématique (1.3). Enfin, dans la quatrième et dernière section, nous présenterons quelques uns des défis contemporains qui sont en débat au sein des communautés, afin d'illustrer les succès et carences propres au mouvement (1.4).

1.1 Ordinateur, matériel et logiciel

1.1.1 L'abstraction du logiciel

L'origine du mot "*computer*" désigne un homme réalisant des opérations mathématiques avec l'aide d'outils mécaniques. Parmi ces outils qui restent les plus célèbres, on trouve le boulier (2700-2300 av J.C.), le mécanisme antikythera (150-100 av J.C.), la règle de calcul et l'astrobale datant de la Renaissance. Les premiers traces de programmabilité, c'est à dire d'automation des opérations de calculs, remontent aussi à l'antiquité avec le théâtre mécanique de Héron d'Alexandrie (1070 ap J.C.) mettant en scène une pièce de 10 minutes à l'aide d'objets animés par un système de cordes et leviers dont l'exécution était décidée à l'avance. C'est en associant ces premiers outils d'automation du calcul avec la notion de programmabilité que l'on peut constituer l'origine du sens donné à l'ordinateur moderne. On peut alors parler de d'ordinateur comme d'une machine traitant des données (calculs) à partir d'instructions données (programmabilité). En français, le terme *computer* se traduisait initialement par "calculateur". C'est un professeur de latin de la Sorbonne, Jacques Perret, qui, mandaté par IBM France, a baptisé l'objet d'*ordinateur* en référence à la figure de l'*ordonnateur* dans la notion d'ordre ecclésiastique de la croyance catholique.

Néanmoins, ce n'est qu'entre 1940 et 1945 qu'ont surgi les premiers ordinateurs semblables à ceux que nous connaissons aujourd'hui. Il s'agissait de grandes machines occupant des pièces entières et consommant l'énergie de centaines de micro-ordinateurs actuels. La révolution informatique des années 1940 est le produit de la fusion entre divers efforts d'explorations théoriques et pratiques [Black, 2002]. En se basant sur la *tradition mathématique* consituée, entre autre, par les oeuvres de Leibniz et Boole, Alan Turing réussit à exposer les bases de l'informatique d'intérêt général dans les années 1930, avec la "Machine Universelle de Turing". Il s'agit d'une représentation abstraite et théorique d'un dispositif informatique. Selon les mots de son propre auteur, "il est possible d'inventer un machine unique pour

informatiser n'importe quelle séquence"[1] [Turing, 1936, p.241].

Cependant, l'inspiration abstraite de Turing s'est réalisée concrêtement par le fait d'une tradition distincte : la *tradition d'ingenieurie*. Cet ensemble d'écoles pratiques a permis, en effet, dans les années 40, que soient construitent les premières machines élèctroniques de calcul. Ainsi, les premiers dispositifs élèctroniques incorporant le système théorique de Turing sont le résultat de la culmination des héritages empiriques d'auteurs comme John Napier (1550-1617), Blaise Pascal (1623-1666), Charles Babbage (1791-1871), entre autres.

Face à ces machines, est apparue la nécessité de les contrôler, de produire des instructions qui pourraient être prescrites. Ainsi, comme conséquence de cette nécessité, est né le troisième élément qui constitue l'informatique moderne : la *tradition de programmation informatique*. Ceci s'est produit notamment dans le cadre de l'effort scientifique réalisé pendant la seconde guerre mondiale, par exemple autour de la machine ENIAC, développée par l'Université de Pennsylvanie.

Servant des intérêts belliqueux, les premières machines informatiques se dédiaient à la rapidité des calculs et, pour cela, étaient construites avec des objectifs particuliers et isolés, de telle manière que l'on entendait alors les notions, aujourd'hui établies, de Logiciel (*Software*) et de Matériel (*Hardware*), comme un seul et unique objet, comme fondamentalement dépendantes l'une de l'autre. Néanmoins, l'idée "d'intérêt général" qui se trouve dans le système de Turing décrivait une plate-forme compatible avec n'importe quel ensemble d'instructions, comme un dispositif générique capable de réaliser des calculs variés et adaptables. En d'autres termes, bien que les racines théoriques de l'informatique aient différencié le logiciel du matériel, les premières réalisations pratiques ont créé des machines qui ne permettaient pas à ces deux objets d'évoluer indépendamment. Ces limitations apparurent avec clarté en élargissant l'usage de ces premières machines au monde de l'entreprise. Selon les mots de Maurice Black : "La clé pour construire un ordinateur utilisable – et, par conséquence, rentable – ne réside pas dans le fait de le dessiner pour réaliser des fonctions ou des calculs spécifiques, mais dans le fait de construire un matériel d'intérêt général qui pourrait être programmé à réaliser n'importe quelle tâche"[2] [Black, 2002, p.40]. Cependant, comme le note l'historien de l'informatique Micheal Mahoney à propos de cette période :

> Nous n'avons pratiquement aucun compte-rendu historique de comment, à partir de années 1950, les gouvernements, les entre-

[1]"it is possible to invent a single machine which can be used to compute any computable sequence"

[2]"the key of making computer usable – and therefore profitable – lay not in designing them to perform specific functions or calculations, but in bulding general-purpose hardware that could easily be programmed to perform any task."

> prises et les industries ont informatisé leurs opérations. En dehors de quelques études avec une dominante sociologique sur les années 1970, la programmation, comme nouvelle activité technique, et les programmateurs, comme nouvelle force de travail, n'ont pas reçu d'attention historique.[3] [Mahoney, 2002, p.92].

Bien qu'il soit difficile de préter attention aux efforts isolés qui ont permis l'abstraction progressive du logiciel comme un objet à part entière, sujet à des évolutions indépendantes de la plate-forme où il est exécuté, on peut néanmoins souligner ici l'un des faits importants de l'histoire de l'informatique, qui a permis qu'émerge le concept moderne de *software*. Dans l'environnement des problématiques visant à construire un *hardware* d'intérêt général, le mathématicien John Von Neumann a décrit [Neumann, 1945] une architecture dont la prétention était d'être entièrement digitale, c'est à dire, que les instructions soient enregistrées dans la mémoire de l'ordinateur et non comme un circuit mécanique spécifique. Ainsi, les instructions pouvaient être transportées d'un ordinateur à un autre. Cette idée de "programme enregistré" a changé le visage de l'informatique moderne et a différencié le matériel du logiciel.

D'un côté, cela a permis que les instructions puissent être développées abstraitement, sans lien physique avec le matériel de la machine et, pour cela, a ouvert les possibilités d'un développement technologique collaboratif. D'un autre côté, ceci a été à l'origine du phénomène de *blackboxing* (boîte noire) des programmes, permettant à ses développeurs de mettre à disposition un produit fini dont le code source serait caché. Ainsi, pour ces deux raisons principales, Maurice Black a affirmé que "l'abstraction du logiciel du matériel est l'aspect le plus important des débuts de l'histoire de la programmation"[4] [Black, 2002, p.49].

Néanmoins, le terme de logiciel a pris du temps à être utilisé par les actants des technologies informatiques. Selon Matthew Füller [Fuller, 2008], le premier usage public de ce mot se trouve dans un article de la revue *American Mathematical Monthy* de 1958 [Tukey, 1958]. Il s'agissait alors de l'idée selon laquelle tout les problèmes mathématiques pouvaient être résolus au sein même des mathématiques, idée qui se rencontre aujourd'hui autour du concept d'algorithme, entendu comme abstraction libérée des détails de l'implémentation. Mais ce n'est qu'en 1968, losque IBM décida, pour échapper

[3]"we have pratically no historical accounts of how, starting in the early 1950's, government, business, and industry put their operations on the computer. Aside from a few studies with a primarily sociological focus in the 1970's, programming as a new technical activity and programmers as a new labor force have received no historical attention."

[4]"Software's abstraction from hardware is the single most important aspect of early programming history."

à une accusation d'*Antitrust* de l'administration américaine, de diviser sa section informatique en deux départements, *Hardware* et *Software*, que le terme de logiciel a pris un bonne partie de son sens commun. Néanmoins, c'est avec le propagation des ordinateurs personnels et la démocratisation de l'accès aux technologies informatiques, que l'objet "logiciel" apparaîtra dans toute sa dimension politique et sociale.

1.1.2 Éléments principaux du matériel et du logiciel

Bien que sa structure ait évolué avec le temps, l'ordinateur est généralement composé des mêmes types d'éléments : une unité centrale de contrôle qui gère les différents composants, une unité arythmétique et logique qui réalise les opérations et une unité de mémoire dans laquelle les données sont enregistrées et lues. L'ordinateur – comme un tout – reçoit des entrées (*input*) et restitue des sorties (*output*). Ce couple *input*/*output* est reproductible à tous les niveaux de la hiérarchie qui compose un ordinateur : entrées et sorties d'un programme, d'un circuit, d'un composant, etc. L'ensemble des éléments physiques d'un ordinateur constituent son matériel (*hardware*). Sa fabrication a été modifiée avec le temps, en fonction d'objectifs de rentabilité, techniques, spatiaux, qui transforment ses origines mécaniques (transistors, tubes, etc.) en circuits miniatures intégrés. Il est intéressant d'évoquer ici que les dernières recherches théoriques à propos de *hardware* présentent des technologies quantiques, chimiques et optiques qui indiquent des performances sans comparaison avec les technologies actuelles les plus avançées.

Cet ensemble matériel gère des données, des programmes, des protocoles, c'est à dire, un ensemble immatériel, désigné par le terme générique de logiciel (*software*). Tous les logiciels sont créés à partir de un ou plusieurs languages de programmation. Il s'agit de langages exclusivement écrits, précis et concis, afin d'éviter toute ambigüité dans les instructions qu'ils formulent. Une fois qu'ils sont interprétés par le *hardware* (compilés), le *software* apparait seulement comme une succession de bits (0 et 1), un flux binaire incompréhensible pour un être humain (Figure 1.1). Bien que les logiciels puissent désigner un nombre illimité d'objets, ils peuvent être classifiés en plusieurs types :

- Le *Système Opérationnel* (SO), comme par exemple, Windows, Unix ou DOS. Il organise et coordonne l'activité et la distribution des ressources finies du matériel.
- Les *bibliothèques*. Un ensemble de "sous-routines" invoquées pour développer des programmes majeurs.
- Les *données*. Il s'agit tout autant de protocoles de transfert (SMTP, FTP, etc), que de formats de fichiers (JPEG, HTML, MPEG, etc).
- Les *applications*. Ce sont les outils bureautiques (MS Office, Open Of-

fice), Internet (navigateur, serveur web), graphique (éditeur d'image, création 3D), audio (Composition musicale, mixage), ingenierie du logiciel (compilateur, debuggeur), jeux, entre autres.

FIG. 1.1 – Code source et format binaire

```
$ echo ‘‘Hello World’’
```

```
01100101011000110110100001101111001000000010001001001000001100101011011000110011011000110111100100000010111001001101100111001000000100010
```

1.2 P.C. et logiciel

1.2.1 le P.C. et l'informatique de masse

Les ordinateurs coûtaient si cher à acheter, utiliser et entretenir que seulement quelques universités, entreprises et organes gouvernementaux en possédaient. Grâce aux évolutions technologiques qui réduirent les coûts de production, l'espace et l'énergie utilisés par chaque unité, ont surgi dans les années 1970 les premiers micro-ordinateurs, ou ordinateurs personnels (*Personnal Computer*, PC), que le terme ordinateur vient aujourd'hui désigner par défaut. Désignant uniquement les ordinateurs de bureau (*Desktop*), récemment, ce terme désigne aussi les ordinateurs portables et les dispositifs réduits (netbook, PDA, etc). Ils sont caractérisés par le fait d'être utilisés par une seule personne à la fois, ce qui les différencie des serveurs.

Il y a un environnement spécifique autour de ces innovations technologiques. Pendant la période de révolution culturelle étatsunienne des années 1970, les campi universitaires (notamment celui de l'Université de Stanford, en Californie) sont le lieu de rencontre des grands noms de la production des PCs, parmi lesquels, on trouve, Steve Jobs et Steve Wozniac (cofondateurs d'Apple), Bill Gates et Paul Graham (Microsoft), Paul Graham, Bill Joy, etc, qui cohabitaient avec divers mouvements sociaux : marxiste, anti-guerre au Vietnam, zen-bouddhiste, écologiste, rock, fiction scientifique, entre autres [Breton, 1990]. Bien que ces récents acteurs des nouvelles technologies ignorent relativemment ces mouvements sociaux, ils restent influencés par une ambiance d'hyper-diversité, alternative, de transgression et d'inno-

vation qui formèrent les bases de ceux qui allait devenir les "Pirates de la Silicon Valley"[5].

D'accord avec Paul Graham, ingénieur informatique et contemporain de cette époque, qui témoigne dans son essai *The Power of a Marginal* : "Le monde n'avait pas encore pris conscience que créer une compagnie d'ordinateurs était du même ordre qu'être artiste ou peintre" [Graham, 2006]. Ainsi, en admettant l'idée que les choses nouvelles et brillantes sont inspirées par les marges d'une société, c'est la coexistence ouverte de beaucoup de marges culturelles dans un même temps et espace qui a permis l'ambiance de liberté et d'audace scientifique propice aux révolutions technologiques.

Les créateurs des ordinateurs personnels n'imaginaient pas le succès qu'auraient leurs modèles technologiques [Negroponte, 1996]. Ils profitèrent de cette situation en bénéficiant d'innovations à moindre coût. Par exemple, la compagnie Xerox, qui a inventé l'interface graphique et la souris, a partagé ses innovations avec Apple, sans recevoir aucun dédommagement. Les dirigeants de Xerox ne croyaient pas en l'utilité de ces outils et ont permis à Apple d'apprendre leurs fonctionements en détail. Ce sont ces mêmes inventions qui ont contribué au succès de l'*Apple II* en 1977 et du PC de manière générale.

Enfin, les premières productions d'ordinateurs personnels étaient réalisées par des groupes réduits, souvent en conditions précaires, ce qui a empêché certains de faire évoluer leurs structures et d'accompagner l'élargissement du marché de consommation. Cependant les grands groupes, comme IBM ou HP, n'ont pas attendu longtemps avant de rentrer dans le marché et proposer des produits équivalents. En plus de s'étendre à un public d'ingénieurs, le PC est apparu comme un moyen d'apporter un meilleure qualité de travail [Toffler, 1985], et c'est dans les bureaux que le PC a été accepté à grande échelle le plus rapidemment. Le modèle de station de travail (*Workstation*), considéré comme une plate-forme de haute productivité, s'est substitué petit à petit aux outils de bureau traditionnels.

1.2.2 Le développement des logiciels et le début de la culture du libre

Le logiciel et l'ensemble des applications qui composent un système n'attirent pas l'attention de l'industrie informatique. À part Microsoft, aucun actant ne développe un système opérationnel à prétention universelle, c'est à dire, cherchant à fonctionner sur n'importe quel matériel. Comme le confie

[5]Titre d'un film de Martin Burke (1999), inspiré par le livre : FREIBERG, Paul et SWAINE, Micheal, *Fire In The Valley*, 2000.

le P.D.G d'Apple à propos de Bill Gates, au cours d'un entretien à l'émission *All Things Digital* : "Il a créé la première compagnie de logiciels avant que quiconque dans cette industrie sache ce qu'est une compagnie de logiciels"[6]. De fait, à cette époque, l'industrie informatique cherchait principalement à vendre et entretenir du matériel. Les logiciels étaient considérés comme accessoires parce que la majorité des utilisateurs les développaient eux-mêmes. C'est l'apparition d'un système multi-tâches (*multitask*) qui a imposé la forme des softwares que nous connaissons aujourd'hui, rendant possible la commercialisation de disques durs et de bandes magnétiques permettant d'enregistrer, modifier et réutiliser des programmes [Muller, 2001].

Une information importante est que beaucoup de systèmes partagés qui fonctionnaient sur les ordinateurs centraux des grandes institutions utilisaient la technologie Unix. Cette technologie fut développée par Ken Thompson et Dennis Ritchie au sein des laboratoires de AT&T (Bell), et est disponible dans sa première version en 1969. Néanmoins, à cause d'une action en justice (1949-1956) pour abus de position dominante et pour éviter toute nouvelle infraction au jugement, le système Unix n'est pas commercialisé. Ce programme est mis sous licence que les institutions pouvaient acheter. Ainsi, le logiciel étant distribué sans service après-vente, ni correction de *bug*, un effort actif de développement se mit en place dans les universités. Le réseau *Usenet* est devenu une plate-forme d'échange d'informations et d'aide à l'utilisation d'Unix et les innovations et corrections de bug circulent rapidemment. En plus d'avoir un rôle de coordinatrice, l'Université de Stanford (Berkeley, CA-EUA) commence à développer sa propre version d'Unix - BSD (*Berkely Software Distribution*) - publiée pour la première fois en 1978 par Bill Joy. Quatre ans après, en 1982, d'autres versions d'Unix sont commercialisées par IBM, HP, DEC, afin d'êtres utilisées sur leurs matériels respectifs. À la même date, Bill Joy quitte l'Université de Berkeley et fonde SUN dont les machines utiliseront BSD 4.2. En 1984, AT&T réussit, après une nouvelle action en justice, à entrer dans le marché de l'informatique comme acteur à part entière et publie pour la première fois une version commerciale d'Unix dont le code-source sera dès lors fermé [Muller, 2001].

Unix a donc été la première proposition suffisamment aboutie ayant permis de penser un système opérationnel universel. Avec un code-source accessible, elle a profité, depuis son origine, des perfectionnements, modifications et adaptations qui ont permis une appropriation étendue, variée et créative. Au-delà de ça, l'influence de ce système opérationnel a dépassé le cadre des constructeurs de serveur. En effet, l'ancêtre de l'Internet, Arpanet, a été organisé et installé à partir de système Unix. Cette technologie, bien que

[6] *All Things Digital*, 30 mai 2007.

propriétaire, a été appropriée par toute une génération d'ingénieurs dans une dynamique constante d'échanges, de divisions et de perfectionnements. C'est probablement là que se trouvent les racines de la philosophie *open-source*, au milieu des ardeurs des révolutions culturelles et technologiques des années 1970 aux États-Unis. Il semble que l'interdiction pour AT&T de commercialiser son produit a laissé un espace temporairement vide dans le processus de radicalisation propriétaire de l'industrie informatique et particulièrement dans l'immense marché qui allait s'ouvrir avec l'ordinateur personnel.

1.3 Logiciel libre et code ouvert : communautés, licences et institutions

1.3.1 Inventer et protéger un logiciel libre

C'est une petite faille dans le système de protection de Unix qui a permis, à court terme, que celui-ci soit approprié, développé et perfectionné par un ensemble d'actants de diverses localités. De cette brève période est restée une tradition d'échange "ouvert" d'informations et, bien que déjà présente dans le milieu de l'informatique, il s'agissait pour ses adeptes, de la protéger pour que ses bénéfices ne soient pas appropriés. Cet intérêt a été rapidemment saisi depuis la permière version d'Unix développée par l'Université de Berkeley et, à cette fin, la Licence BSD a été créée. Cette déclaration contractuelle oblige les développeurs et les utilisateurs à mentionner les noms des auteurs dudit programme, elle exclue toute responsabilité en cas de dysfonctionnement, le code-source est ouvert, mais il peut-être fermé, pour une utilisation commerciale par exemple.

Á partir de 1984, la Fondation du Logiciel Libre (FSF - *Free Software Fondation*), fondée par Richard Stallman, va articuler en termes idéologiques et politiques les défis d'un logiciel qui veut être désigné de "libre". L'idée est que chaque ligne de code est la parcelle d'une information qui, étant suivie, partagée et discutée, ira mieux s'adapter. Nous pouvons observer que les métaphores ne manquent pas aux discours de la fondation, les recettes de cuisine, par exemple : en découvrant que battre des oeufs et les jeter dans une poêle permet de faire une omelette ; pourquoi alors devrions-nous limiter notre liberté de partager cette astuce avec notre voisin et notre curiosité à développer des recettes dérivées ? Avec cette simple question, Richard Stallman ne veut pas défendre un logiciel gratuit, comme peut le laisser entendre le mot anglais *free* ("*free as free beer*"), mais plutôt un modèle de développement ("*free as free society*"). Pour cela, en français comme en portugais, ce type de logiciel est dit "libre/*livre*" et non "gratuit/*gratuito*". De même, c'est

pour cette raison que l'appellation la plus utilisée par les institutions internationales aditionnent au sigle anglais le terme *libre* de la langue espagnole et française :*Free/libre and Open-Source Software* (F/LOSS).

Pour servir ces objectifs, Richard Stallman écrit la Licence Publique Générale (GPL - *General Public License*) qui inspire le mouvement du logiciel libre. Un code sous licence GPL est ouvert et modifiable, et doit rester ainsi. Aucune restriction à ces conditions de bases ne peut être additionnée. En complément de son intérêt juridique, la GPL véhicule une idéologie symbolisée par les "quatres libertés"[7] : la liberté d'éxécuter le programme, dans n'importe quel but ; la liberté d'étudier comment le programme fonctionne et de l'adapter à ses nécessités ; la liberté de le redistribuer ; la liberté de l'améliorer et de le partager avec la communauté.

Protégée par cette licence, la FSF va initier un projet ambitieux, visant à offrir un système opérationnel et des applications performantes entièrement libres. C'est ainsi que, dans le cadre du projet GNU (acronyme de *GNU is Not Unix*), des centaines de programmeurs des régions développées du monde ont développé des applications semblables à celles offertes par Unix. Plusieurs des applications créées à cette époque restent jusqu'à aujourd'hui connues et performantes (notamment le compilateur GCC). En dépit de ces nombreuses années de travail, il n'est pas encore disponible un noyau (*kernel*) qui permette le fonctionnement de toutes ces applications sur un système unique. En 1991, Linus Torvalds, programmeur finlandais, publie le code-source de son *kernel* "Linux" qui achève l'oeuvre du projet GNU et réalise le premier système d'exploitation universel sous licence GPL : GNU/Linux.

1.3.2 Logiciel Libre *et/ou* ouvert

Alors que le mouvement du logiciel libre évolue encore dans un environnement peu connu, des voix se lèvent en son sein pour une ouverture aux partenariats avec des institutitons privées. De fait, plusieurs communautés ont proclamé que le succès de Linux n'était pas lié à sa nature "libre" mais aux avantages et à la stabilité offerts par l'accès au code-source et son modèle de développment collaboratif. En 1998, le mouvement du logiciel libre connaît une scission institutionnelle. Après 7 ans de vie, le système Linux n'était plus marginal et commençait à apparaître comme une alternative crédible pour les entreprises : Intel s'associe à Linux International, IBM construit du matériel pour être utilisé avec Linux, Netscape publie le code-source de son navigateur sur mozilla.org, les codes-sources de Staroffice (ancêtre de Open

[7]Référence aux quatres libertés proclamées par le Président des EUA, Franklin D. Roosevelt lors de son discours au congrès Américain, le 06/01/1941.

Office) et Solaris (système d'exploitation de SUN) sont, eux aussi, publiés. Beaucoup de voix se retrouvent autour d'événements comme la publication de l'essai de Eric Raymond, *The Cathedral and the Bazaar* [Raymond, 1997], et le *Open-Source Summit* réalisé par les éditions *O'Reilly*. L'idée développée lors de ces évènements est d'écarter les prétentions idéologiques de la FSF et la défense du droit à la co-création/modification du code produit. C'est autour de ces idées que se crée ainsi le mouvement du logiciel *open-source* (ouvert), comme philosophie pragmatique des principes qui firent le mouvement du logiciel libre, appuyée en partie par des institutions privées (IBM, Intel, O'Reilly, SUN, ReHat, etc.).

Cette dénomination de "ouvert" insiste sur l'ouverture du code-source et met de côté les engagements politiques et idéologiques, qui seront, dès lors, les revendications quasi-exclusives de la FSF et de ses projets connexes. On rassemble en général ces deux mouvements sous le sigle de *Free/Libre and Open Source Software* (FOSS, F/LOSS) et en portugais, "Logiciel libre et à code ouvert" (SL/CA - *Software Livre e de Código Aberto*). Pour ses propres acteurs, les noms de "mouvement *open-source*" ou de mouvement "du logiciel libre" invoquent des ambitions bien différentes qui refusent d'être assimilées les unes – idéologiques – avec les autres – pragmatiques. Cependant, si les défenseurs de l'*open-source* font librement référence aux icônes de l'aile "libre" du mouvement, l'inverse n'est pas bienvenu, et la simple prononciation de "*open-source*" sur les canaux IRC du projet GNU ou de la FSF[8] reçoit un "on ne fait pas d'*open-source* ici !".

Fig. 1.2 – Appellations du mouvement du logiciel libre

FOSS, F/LOSS, SL/CA

Branche politico-idéologique	Branche pragmatique
Free Software, Logiciel libre	Logiciel ouvert *Open-Source Software (OSS)*
Exemplo de comunidades : GNU	Exemplo de comunidades : BSD, Mozilla

1.3.3 Licences et institutions

Depuis le début du mouvement FOSS, et plus encore depuis la scission explicite de 1998, se sont multipliées les licences pour protéger les travaux

[8]Canaux #gnu & #fsf serveur Freenode (irc.freenode.net)

des communautés. Dans une certaine mesure, ces licences protègent toutes l'accès au code-source mais elles se différencient par les restrictions qu'elles apportent aux quatre libertés fondamentales du projet GNU. Si quelques uns des projets qui sont financés par des entreprises restent sous licence GPL, beaucoup d'institutions, privées comme publiques, préfèrent développer des licences propres, s'écartant alors des libertés déclarées par la FSF. Souvent, les licences spécifiques sont créées pour un projet unique, pour cela on en dénombre des centaines dont les différences sont de permettre, ou non, l'utilisation avec un logiciel propriétaire, l'accès aux modifications, la publicité, les droits particuliers du directeur du projet. Selon ces critères, voici un tableau comparatif de quelques unes des licences les plus connues[9] :

FIG. 1.3 – Comparaison entre quelques-unes des principales licences libres

Licence	Utilisation avec un Logiciel Commercial	Accès aux modifications à tous	Publiable sous certaine conditions	Présence de droits particuliers réservés au propriétaire de la licence
GPL (*General Public License*)	Non	Oui	Non	Non
LGPL (*Lesser General Public License*)	Oui	Oui	Non	Non
BSD (*Berkeley Software Distribution*)	Oui	Non	Non	Non
NPL (*Netscape Public License*)	Oui	Non	Non	Oui
Domaine public	Oui	Non	Oui	Non

Certaines des licences créées pour protéger un projet spécifique sont réutilisées par des projets indépendants ; de fait, l'aura que gagnent certains projets ouverts leur donne une influence propre au sein du monde du logiciel libre. Ainsi, le projet Apache, avec plus de 50% des parts de marché comme serveur Web, a créé une licence propre (*Apache Public License*) qui a été

[9]Tableau copié du livre : *Tribune Libre : Ténors de l'Informatrique Libre*, editions O'Reilly, p.200.

copiée par de nombreux autres projets. Cette licence est compatible avec la GPL depuis sa version 2.0 (Janvier 2004) et est utilisée notamment par Google pour certains projets ouverts. Pour illustrer une situation différente, la fondation Mozilla (projet Firefox notamment) propose une licence incompatible avec la GPL (*Mozilla Pulic License*) mais tout de même reprise par SUN (*SUN Public license*).

Cette situation peut être une entrave pour le monde du développement logiciel et à sa dynamique. En effet, l'idée de la GPL est de permettre à un ensemble de codes d'être compatibles, assemblables, intégrables, assimilables... D'un côté, le fait pour un code d'être protégé par la GPL assure qu'il ne peut être reproduit/modifié seulement sous ce même règlement. Pour cela, certains ont désigné la GPL de Virus Public Général (GPV - *General Public Virus*)[10] car elle crée et étend un monopole unique. De l'autre côté, avec la multiplication des licences, nous sommes confrontés à de nouveaux problèmes juridiques :

> Quel statut doit adopter un logiciel contenant du code ouvert par différentes licences – par exemple, un programme qui intègre à la fois du code MacOS d'Apple, X server et Mozilla pour donner une version libre de Netscape Navigator ? Si de plus, la couche utilisateur de ce nouveau programme fictif utilise la bibliothèque Qt de Troll Tech, une troisième licence entre en scène. Les trois licences entraînent des limitations différentes. [Muller, 2001, p.16].

Nous pouvons dire que le domaine du développement logiciel se trouve ralenti par la définition de stratégies juridiques coûteuses et souvent laborieuses. Au-delà des différences sur les externalités juridiques du modèle ouvert/libre, on trouve un processus tout aussi obscur au niveau institutionnel de ces définitions. De fait, il y a une concurrence de légitimité à définir la classification de ces licences, pour définir ce qui est libre/ouvert de ce qui ne l'est pas. La FSF maintient un répertoire actualisé[11] de toutes les licences qui se veulent libres et les classe en fonction de leur compatibilité avec les différentes versions de la GPL, c'est à dire, sa propre définition du "libre". La *Open Source Initiative*, fondée par Eric Raymond en 1998, alors appuyé par l'entreprise Netscape, maintient sa propre banque de données[12], chacune des licences étant soumise à un processus d'approbation répondant à une "définition de l'*Open-Source*" soulignant l'ouverture et le modèle de développement. De la même manière, l'ONU, par l'intermédiaire de ses programmes de développement (UNDP - *United Nations Developement Programs*) a fondé une

[10] Expression pejorative commune parmi les premières critiques faites à la (L)GPL.

[11] `http://www.gnu.org/licenses/license-list.html#SoftwareLicenses`

[12] `http://www.opensource.org/licenses`

institution ayant pour but de définir et développer le logiciel libre et ouvert.

Ainsi, le *International Open-Source Network* (IOSN)[13], qui agit particulièrement dans la région Asie-Pacifique, lui aussi définit et classe les projets et leurs licences au bénéfice d'une conception hybride (liberté/modèle de développement) tournée vers les problématiques d'inclusion/exclusion digitale et de gouvernement électronique. Les gouvernemnts nationaux, particulièrement ceux de la France, du Brésil, de la Chine et d'Israel, se concentrent sur les cas d'adoption des sytèmes ouverts dans l'administration, favorisant les critères de gratuité (réduction des coûts) et d'accès au code-source (contrôle/sécurité). En Europe, c'est le cas de *Interoperable Delivery of European eGovernment Services* (IDABC) et de l'*Open-Source Observatory and Repository* (OSOR). Au Brésil, les initiatives semblables étaient inexistantes avant l'an 2000 et ne restaient qu'à des niveaux locaux, comme c'est le cas des télécentres (*telecentros*) qui ont privilégié peu à peu l'usage de Linux. Le Logiciel Libre est devenu une priorité au niveau fédéral après un décret présidentiel de 2003[14] attribuant au Comité Exécutif du Gouvernement Électronique (CEGE - *Comitê Executivo do Governo Eletrônico*)[15] "l'implementation du logiciel libre". Dans ce cas, le logiciel libre est considéré comme un recours stratégique, une option technologique favorisant, à court et long terme, une autonomie et une appropriation indépendante des outils technologiques utilisés de manière transversale dans toutes les réalisations techniques du Gouvernement Électronique. Selon Tulio Vianna [Vianna, 2006], le logiciel libre est un moyen de garantir concrètement le droit économique au développement technologique. Ce droit souligne tout autant l'idée d'une autonomie scientifique nationale, que celle d'une démocratisation de l'accès aux nouvelles technologies. Cela serait, selon les mots de l'auteur, une arme "contre la colonisation technologique des États-Unis" et un moyen de garantir une formation plus indépendante des élites.

1.3.4 Postulat de Définition

Nous voyons que les communautés, institutions et licences qui s'opposent quant à la définition des objectifs et des moyens du "mouvement du logiciel libre", créent une concurrence autour de la légitimité à définir ce mouvement.

[13] `http://www.iosn.net` Pour les discussions par rapport à la reglementation des projets libres, voir : `http://www.iosn.net/licensing/foss-licensing-primer/`

[14] Décret Présidentiel du 29 décembre 2003

[15] Le CEGE a été créé le 18 octobre 2000 dans le cadre du Conseil de Gouvernement de la Présidence de la République, avec l'objectif de "formuler des politiques, établir des directrices, coordonner et articuler les actions d'implémentation du Gouvernemnt Électronique".

Pour le sociologue, il s'agit de traiter de manière systématique un objet dont la nature est fluide et évolutive, comme c'est souvent le cas à propos de mouvements sociaux, par exemple, dans les analyses du mouvement alter-globalisation [Kavada, 2003].

Dans notre cas, les analyses de réseaux proposées par Arturo Escobar [Escobar, 2003] semblent faire sens avec la structure du bas par le haut (*bottom-up system*) et le haut degré d'auto-gestion qui se trouve aux alentours du logiciel libre et de ses communautés. En considérant la métaphore de la "maille" (*meshwork*) au lieu de celle de "réseau" (*network*), l'auteur permet l'observation d'une grande héterogénéité au sein même du mouvement social.

De cette manière, nous pouvons inclure la plupart des contradictions soulignées par les différents actants du mouvement. En prenant 3 exemples emblématiques, ceux des communautés BSD, GNU et Mozilla, nous pouvons observer que la première - BSD - n'est pas liée à une institution, mais à une licence et permet l'utilisation commerciale et fermée de son code. Pour cela, elle est exclue de la vision du libre de la communauté GNU, laquelle est liée à une institutiton (FSF) et à une licence (GPL). Les logiciels GNU sont reconnus comme "libres" par les deux autres communautés commentées ici. La communauté Mozilla est exclue par les critères de la FSF, mais ne permet pas, néanmoins, la fermeture du code de ses produits. Du point de vue de GNU, seul GNU est libre. Du point de vue de BSD, toutes les trois sont libres. Du point de vue de Mozilla, seuls GNU et Mozilla sont libres. De plus, par rapport à la dénomination *open-source* : GNU ne veut pas se considérer comme *open-source*, alors que BSD et Mozilla considèrent les trois communautés comme *open-source*.

Dans cet environnement d'opposition, postuler que le mouvement du logiciel libre est composé de tous les acteurs qui prétendent faire partie du monde du "libre" ou "*open-source*" paraît un moyen efficace de capturer le mouvement dans toute sa complexité, restant capable de révéler les nuances identitaires des actants, lors de l'observation d'un contexte particulier.

1.4 Le succès du code ouvert et ses défis contemporains

Le changement qui est intervenu en 1998 avec la scission explicite entre les mondes du libre et de l'*open-source* a comme conséquence majeure l'entrée des entreprises et l'attribution de budget jusqu'alors inespéré. De telles attitudes sont des moyens importants qui sont établis par des entreprises

comme IBM ou SUN pour adapter les logiciels à leurs nécessités. C'est à cette époque que vont survenir des victoires décisives du Libre, quant à son influence. Par exemple, en 1998, IBM participe au projet Apache, un serveur Web, qui montre une certaine stabilité dans son développement, alors que ses équivalents propriétaires se perdaient dans des réorientations constantes. Aujourd'hui, Apache est une des conquêtes les plus significatives du Libre, étant utilisé par la majorité des serveurs web dans le monde (70%[16]). À cette époque, IBM a investi un million de dollars dans Linux pour l'installer sur ses serveurs. Le système opérationnel FreeBSD a profité des investissements de Apple, Google ; d'autres projets connaissent un succès notable : DNS et BIND, Sendmail, Samba, Perl, Python, Tcl/Tk.

Nous pouvons observer que les applications libres qui ont connu du succès sont relativement "fondamentales", c'est à dire, des systèmes et des programmes pour des serveurs, protocoles de réseaux ou de fichiers, des langages de programmation, etc. Dans cet environnement, le monde du *Desktop*, de l'ordinateur de bureau, de l'utilisateur final (*end-user*) est encore à conquérir. De manière générale, une machine *Desktop* est composée d'un système opérationnel auquel on additionne quelques éléments essentiels. Avec 90% des parts de marché[17], Microsoft détient le quasi-monopole des systèmes opérationnels utilisés sur les ordinateurs personnels (*Microsoft Windows*) et réduit ainsi l'usage de Linux à une minorité d'utilisateurs avancés (0,90%). Cependant, depuis 2005, Mark Shuttleworth finance, à travers la société de service *Canonical Ltd.* (Royaume-Uni), une version de Linux (Ubuntu) ayant pour objectif d'être intuitive et dédiée aux utilisateurs communs. Cette distribution de Linux semble développer sa présence dans le monde du *Desktop*, notamment en étant adoptée par de grandes institutions (Wikimedia, Amazon.com, Gendarmerie française). Open Office (OOo), alternative à Microsoft Office, est un projet ancien qui bénéficie de l'appui de SUN et se trouve par défaut dans toute les distributions de Linux, tout en étant disponible aux utilisateurs de systèmes Apple ou Microsoft. Une des plus grandes conquêtes du "libre" dans le monde du *Desktop* est le navigateur Web Mozilla Firefox (22%[18] du marché), qui réussit à concurrencer son équivalent propriétaire (Internet Explorer - 66%), en offrant plus de sécurité et d'adaptabilité.

Dans une observation plus générale et économique du phénomène de l'*open-source*, on note la nécessité de trouver des modèles de financements locaux qui permettent une rétribution aux développeurs et volontaires. Le début du projet GNU décrivait la participation comme un hobby rétribué

[16]Fonte : `http://www.securityspace.com/s_survey/data/200905/index.html`
[17]Fonte : `http://marketshare.hitslink.com/report.aspx?qprid=8`
[18]Source : `http://marketshare.hitslink.com/report.aspx?qprid=0`

par un emploi lié au domaine de la participation volontaire. Ainsi les développeurs du système GNU/Linux utilisaient une partie de leur temps de travail comme analyste système, administrateur de réseau ou programmeur, à développer des projets qui allaient donner un retour pour les entreprises employeuses. Avec l'influence de la culture *open source* à d'autres domaines, l'équilibre entre volontarisme et rétribution n'était pas aussi bien défini. Par exemple, un projet comme Wikpédia, qui fait partie des 10 sites les plus visités au monde[19] n'emploie que 23 salariés[20]. Dans ce sens, le mouvement du logiciel libre contribue au "Travail du consommateur" [Dujarier, 2008] faisant que l'utilisateur-développeur crée une valeur pour l'actionnaire, ceci sans coût. Dans une étude sur la nouvelle "économie collaborative" (*Wikinomics*), le groupe d'étude *New Paradigm* a révélé que la pièce maîtresse de ce nouvel échiquier est l'étape de contact entre le développeur et le projet auquel il va participer. Ainsi, des plates-formes offrant ou obligeant à une rétribution garantissent un modèle économique transparent, alors que des projets plus informels ne peuvent pas garantir un retour pour l'utilisateur-développeur [Tapscott and Williams, 2008].

L'ouverture des communautés FOSS aux investissements privés s'est réalisée au profit d'une vision pragmatique du mouvement, c'est à dire, mettant en lumière ses qualités comme modèle de développement. Les entreprises privées ont rapidemment compris leurs intérêts à favoriser la curiosité créative de leurs clients/utilisateurs, tout comme les efforts techniques des développeurs pour adapter les produits à leurs nécessités. De cette manière, le mouvement reste éloigné du concept de *copyleft* et de la philosophie du projet GNU et les exemples qui suivent viennent illustrer les récentes conquêtes de l'aile pragmatique du mouvement du logiciel libre et les problèmes que cela peut engendrer.

1.4.1 Le kit de développement logiciel (SDK - *Software Development Kit*)

Le SDK (*Software Development Kit*) est une plate-forme de travail mise à disposition par le producteur d'une technologie pour que les développeurs puissent créer/modifier des applications. Il s'agit d'un modèle très utilisé pour divers produits : matériel, SO, Jeux. Ce modèle se trouve tout autant pour des produits libres (Qt, JAva, Android) que pour des produits propriétaires (Flash, iPhone, Microsoft Platform) et réutilise les principes caractéristiques du modèle de développement collaboratif. Qu'il offre, ou non, l'accès au code-

[19]Source : `http://www.alexa.com/topsites`

[20]Source : `http://en.wikipedia.org/wiki/Wikimedia\_Foundation`

source, le programme en question profite de l'effort collectif des utilisateurs-développeurs.

Ceci pour être une réduction forte du concept d'*open source* en faveur d'une vision très rentable pour l'entreprise productrice de la technologie. De cette manière, on peut réduire au minimum le partage des informations avec la communauté tout en profitant des avantages d'une plate-forme collaborative. Ce débat s'est montré très présent lors du lancement de la plate-forme de développement pour l'iPhone d'Apple. Le SDK est téléchargeable et utilisable librement, mais toute publication de code nécessitait un enregistrement au programme officiel. Á la suite de cette formalité, les développeurs - même étant volontaires - étaient soumis à une clause de secret professionnel. Malgré le grand succès du dispositif et de son kit de développement, les protestations ont été telles contre la politique d'Apple que l'obligation au secret a été levée, au moins entre développeurs [Willis, 2008].

De plus, le kit de développement de logiciel permet aux compagnies de faciliter la collaboration des utilisateurs-développeurs, en limitant comme elles le souhaitent les compensations pour la communauté. Pour cela, le SDK est une menace directe au mouvement du logiciel libre et une restriction certaine au modèle de développement *open-source* en ce que ses termes, limitations et objectifs sont entièrement établis par le société émétrice.

1.4.2 L'informatique dans les nuages

Dans le domaine de l'administration de réseaux et de la gestion des informations, deux modèles s'opposent : favoriser le client (décentraliser l'information) ou le serveur (centraliser l'information). Les détenteurs d'informations primaires - ingenieurs, développeurs, professionnels des TI - décrivent un phénomène cyclique à l'échelle de l'administration des réseaux privés (entreprises, collectivités). Cependant, à l'échelle mondiale, le phénomène semble bien plus constant. En effet, le développement progressif d'applications Web (*Web-based*) tend à centraliser l'information sur quelques serveurs de multinationales. Le meilleur exemple, dans ce cas, sont les services en ligne de Google. De fait, avec un simple navigateur, un utilisateur peut utiliser, modifier et stocker ses photos (Picasa, Flickr), vidéos (Youtube), blogs (Wordpress, Blogger), textes, feuilles de calculs, présentations (Zoho, Google Documents), emails (Yahoo, Gmail), itinéraires (Google Maps), favoris et notes (Delicious, Google Note), contacts et agendas (Google Agenda), flux d'informations (Google Reader), créer et héberger ses sites (Google Sites), entre autres. Cet ensemble d'applications dépasse déjà tout ce dont peut avoir besoin un utilisateur commun connecté à Internet. Cette situation permet d'affirmer que le navigateur pourrait devenir l'unique moyen nécessaire pour

accéder aux services que les ordinateurs offrent aujourd'hui. Dans ce sens :

> "Je pense pour ma part que les applications sont en train de passer au modèle Web, dans lequel le navigateur devient le principal, voire l'unique, logiciel. On a, chez Mozilla, une boutade : "le système d'exploitation n'est qu'un ensemble de drivers [programmes faisant fonctionner les périphériques de l'ordinateur, comme la carte graphique] servant à faire tourner le navigateur". Dans cette approche, Windows ne vaut pas mieux que Linux ou que Mac. Il est en arrière-plan, on l'oublie. Quant à l'application bureautique, elle est "dans le nuage", sur Internet." [Nitot, 2008]

Cette évolution est désignée par l'expression de "l'informatique dans les nuages" (*Cloud Computing*) et est en passe de devenir un fait notable et stable de l'histoire de la microinformatique. De plus son avenir est encore plus assuré avec la multiplication des petits clients (*thin-client*) comme les smartphone ou les netbooks, dont les capacités matérielles ne permettent pas d'accueillir des applications locales traditionnelles.

Bien que Linux ait eu un certain succès sur ces plateformes portables, le modèle d'applications *web-based* menace les conquêtes déjà obsolètes du Libre en terme d'applications locales (Gimp, Open Office) en offrant des environnements intuitifs, accessible de n'importe quel endroit connecté, mais dont le code est fermé.

1.4.3 Google

Les acteurs privés ont eu un rôle déterminant dans le développement des logiciels ouverts et c'est grâce a ces investissements que beaucoup d'entre eux ont gagné en crédibilité. Des entreprises comme IBM (Linux), SUN (Open Office, BSD, Virtual Box), Red Hat (Fedora), Canonical (Ubuntu) ont participé de cet effort en fonction de leur intérêts et stratégies respectifs, mais aucune n'a contribué au phénomène, de manière globale, comme le fait Google depuis sa création.

Créée en 1996 comme simple moteur de recherche, l'algorithme utilisé pour les recherches a donné des résultats inespérés et a permis à l'entreprise de croître rapidement et de devenir, en à peine 10 ans, un des acteurs principaux des nouvelles technologies. Grâce à un système de publicité discrète et dédiée par l'examen des informations laissées par chaque utilisateurs, l'entreprise détient maintenant un capital financier et une influence considérable. Selon la *Harvard Business Review* [Iyer and Davenport, 2008], ce sont des milliers d'expérimentations quotidiennes qui sont réalisées sur la plus grande base de données sur la manière dont les personnes cherchent, trouvent et

utilisent les informations. De plus, Google détient le plus grand nombre de profils et d'informations sur les utilisateurs d'internet dans le monde, ce qui pousse l'entreprise même à se présenter de la sorte : "Chez Google, notre mission est d'organiser toute l'information du monde"[21].

Une telle influence et capacité d'investissement ont beaucoup contribué au développement et à la création de nombreux projets *open source*. Google libère des employés pour faire avancer certains projets clés, en plus d'organiser un grand réseau de collaboration dans lequel des étudiants du monde entier peuvent candidater pour participer à des projets ouverts aidés par l'entreprise (*Google Summer of Code*). De même, certains code-sources de ses propres projets sont mis à disposition (ex : Android, Chrome).

Enfin, les opinions exprimées sur les contributions de Google à la communauté du libre sont radicalement divisées et illustrent bien les oppositions entre les adeptes de l'*open source* et du *free*. Pour les plus radicaux et engagés à préserver les origines éthiques et idéologiques du mouvement, "*Google is devil*"[22] (Google est le diable) et enferme le mouvement dans les compromis déjà réalisés par l'aile pragmatique du mouvement, mais, cette fois, sous le contrôle d'un monopole d'entreprise. Pour les pragmatiques du mouvement, le soutien de Google est précieux et les transgressions faites à la GPL sont déjà monnaie courante. Cependant, tous restent préoccupés en observant certaines contradictions allant dans le sens d'une stratégie de contrôle et de monopole. Du point de vue de l'entreprise, il semble que la participation inédite qu'ils offrent aux communautés ouvertes est tout autant une clé de leur succès, qu'un risque de voir cette force de travail, qu'il ne contrôle pas, disparaitre ou changer les termes de sa participation [Fabernovel, 2008, 2009].

[21]VARDA, Kenton, Protocol Buffers : Google's data interchange format, Google code blog, 07/07/2008.

[22]Sujet d'une liste de mails du groupe *Digital Freedom Global Activists* (DFGA – binaryfreedom.info)

Chapitre 2

Aspects sociaux du mouvement du logiciel libre

Ce chapitre a pour but de présenter les éléments politiques et sociaux du mouvement du Libre présent dans la littérature universitaire développée à son propos et d'en proposer une typologie. Après avoir présenté les termes utilisés pour désigner les actants du mouvement au sein des communautés et dans les sens commun et universitaire (2.1.1), nous aborderons le mouvement à travers le concept d'éthique hacker (2.1.3), entendu dans un contexte théorique de construction sociale de l'éthique (2.1.2). Une critique constructive de ce concept permet de réinterpréter les significations culturelles du hacking (2.2) en usant de la notion "d'agnosticisme politique" (2.2.1). De cette manière, l'intéret est porté sur la passion pour l'information comme matrice minimale des interactions de nature politique des utilisateurs-développeurs, permettant alors de construire une typologie de celles-ci à partir des travaux de Gabriella Coleman (2.2.2).

Ainsi, se construit un paradigme d'interprétation de l'activité de l'utilisateur-développeur de logicel libre à travers l'idée de pragmatiques de la programmation (2.3). Après avoir présenté l'acte de programmer dans sa dimension artistique et régulatrice (2.3.1) nous pourrons utiliser des pragmatiques de la programmation comme une interprétation sociopolitique d'un concept linguistique, permettant de comprendre la relation des utilisateurs-développeurs avec leurs politiques informelles et l'information (2.3.2).

2.1 L'Éthique Hacker

2.1.1 Hackers, utilisateurs-développeurs, geeks et nerds

De nombreux termes désignent les personnes qu'on trouve autour du logiciel libre. Cependant, des significations supplémentaires de chacun d'entre eux pointent des catégories plus spécifiques ou générales d'individus. Pour cette raison, nous présenterons, dans le détail, quatre des appellations qui ont été rencontrées fréquemment au long de notre recherche bibliographique et de terrain.

Hacker

"Hacker" est probablement le terme le plus romancé à propos de la culture informatique. L'industrie cinématographique hollywoodiene a, d'ailleurs, déjà profité des recours cognitifs qui se trouvre autour de ce domaine : un "ego illimité" exprimé dans un monde de 0 et 1, qui affronte un réalité manipulée, tronquée, décevante, limitée...La trilogie des frères Wachowski[1], *Cypher*[2], *Pi*[3], développent cette même idée, sans compter les innombrables productions de second plan, dont la liste ne peut prétendre à l'exhaustivité[4].

Ce terme, Hacker, est presque toujours présenté comme dérivé du terme anglais *Hijacker*, pirate. Moins fréquemment, on souligne sa relation étymologique avec le verbe "couper" des langues germaniques et aussi anglo-saxonnes. Alors que l'étimologie anglaise fait référence au sens très médiatique d'un bandit exploitant les systèmes d'informations et les réseaux (*Cracker*), l'étimologie germanique ouvre des horizons plus amples de significations. De fait, il s'agit de "couper" les chemins déja connus pour trouver des astuces, des solutions, des hypothèses pour un problème, qui soient originales, non-évidentes et élégantes. C'est dans ce sens que le terme *hack* a été utilisé la première fois, au sein des laboratoires du club du *Tech Model Railroad* (TMRC) du MIT, dans les années 1950. Ainsi, on nommait "*hack*" les modifications intelligentes qui se faisaient aux relais électriques. De même, dans les années 1960, on faisait référence à l'acte d'hacker (*hacking*) au sein du laboratoire d'intelligence artificielle de la même université, où a été créé le projet GNU. On désignait alors le fait d'aller chercher le code source pour

[1] *The Matrix* (1999), *The Matrix Reloaded* (2003), *The Matrix Revolutions* (2003), de Larry and Andy Wachowsky.

[2] Vicenzo Natali, *Cypher* (2002)

[3] Darren Aronofsky, *Pi* (1998)

[4] *Wargames*, de John Badham (1983) ; *Hackers*, de Iain Softley (1995) ; *Pay Check*, de John Woo (2003) ; *Live Free or die hard*, de Len Wiseman (2007), entre autres.

améliorer et adapter un logiciel à ses propres besoins. Dans ce sens, "hacker" est particulièrement lié avec le mouvement du logiciel libre. Les entreprises brésiliennes d'informatique comme Google Brasil, Microsoft Brasil, IBM Brasil, ont toujours fait référence au "déchiffreur" (*Decifrador*), en respect pour la langue portugaise.

Geek et Nerd

Geek est une expression anglaise dérivée du mot *geck*, idiot (*fool*) et étrange/marginal (*freak*). De même, le terme de geek a des origines avec le mot *geck* du germanique ancien, signifiant fou et avec les dialectes français, *gicque*, signifiant "fou de carnaval"[5]. Il s'agit d'un personne étrange, ou au moins étrangère, particulièrement une étant perçue par les autres comme étant obsédée par une plusieurs choses, parmi lesquelles, l'intellectualité, l'electronique"[6]. Nous pouvons qualifier la culture *geek* comme un mouvement culturel populaire typique des sociétés contemporaines de l'information. De plus, les geeks sont qualifiés comme ayant des intérêts exhaustifs et créatifs sur des sujets liés aux nouvelles technologies [Konzack, 2006]. Pour cela, le logiciel libre peut être considéré comme un domaine de la culture geek et les individus y participant, comme des geeks. Ainsi, l'acte de programmer s'associe parfaitement aux exigences d'exhaustivité, de créativité et de changement présentes dans la culture geek.

L'anthropologue Christopher Kelty, dans son étude des significations culturelles du *free software*, fait constamment référence aux *geeks* comme étant les actants principaux du mouvement [Kelty, 2008]. Pour l'auteur, il s'agit d'une convergence philosophique du côté du transhumanisme. Le transhumanisme est une modalité de croyance en la ligne du temps du progrès technique, laquelle permettrait de dépasser le corps humain. Pour les transhumanistes, l'intervention technique a un rôle spécifique qui la différencie de l'intervention politique, culturelle ou sociale. Dans ce contexte, il n'y a pas de rhétorique, mais une hypothétique "vérité pure" du "ça fonctionne!". Á l'extrême et pour utiliser une métaphore propre au monde de la programmation : "les mauvaises idées ne se compileront pas"[7].

Bien qu'on trouve une connotation de "bizarre" autour du terme *geek*, il semble que la connotation péjorative s'attache au mot *nerd*. De fait, *nerd* désigne une personne avec de hautes capacités intelectuelles et ayant des intérêts hors des normes sociales communes. Ainsi, on se concentre davantage sur

[5]Source : Oxford American Dictionary

[6]Dictionary.com

[7]Un programme est "compilé" quand une série de commande et d'instructions deviennent une application exécutable.

les signifiants de deviance, d'étrange, de hors-normes. Dans le sens commun, il semble que la figure du *geek* se construise dans le contexte d'un spécialiste autodidacte, alors que celle du *nerd* désignerait un adolescent boutonneux. Néanmoins, ces différences semblent apparaître quand on cherche à créer des catégories, mais le conflit geek-nerd n'a pas de sens pour ses acteurs, et chacun peut nommer l'autre de l'une des deux nominations sans avoir une identité très différente. Ce terme peut encore être utilisé, comme l'est geek, pour désigner les acteurs du mouvement du logiciel libre. Par exemple, le *nerd* et l'idéal-type du hacker utilisé par Paul Graham dans *Hackers and Painters* [Graham, 2004].

Enfin, le terme d'utilisateur-développeur est le seul revu ici qui est spécifique aux communautés du logiciel libre. Il désigne des individus qui, autour de ces communautés, utilisent des logiciels libres et par cette utilisation en arrivent à aider au développement du propre software. Cette aide peut être minimale ou presque symbolique, en acceptant de renvoyer des "*crash report*"[8] ou de répondre à des questionnaires. Il peut aussi s'agir de travaux plus majeurs, comme la réalisation de documentations, la programmation de modules additionnels ou du développement du propre software.

La figure des utilisateur-développeurs est très présente dans les projets libres. Ils constituent la "communauté" qui défini ce modèle de développement comme collaboratif. C'est probablement par la reproduction de cette figure à d'autres domaines de production que se perçoit le mieux l'influence que peut avoir le mouvement du logiciel libre hors de ses frontières. Il s'agit, par exemple, des rôles distribués de "consommateur-producteur" de l'économie collaborative (Wikinomia) et de "lecteur-rédacteur" dans le production de contenu dans le Web 2.0, qui sont intimmement liés à l'imaginaire du logiciel libre et à la figure de l'utilisateur-développeur [Tapscott and Williams, 2008].

2.1.2 La construction sociale de l'éthique

Une des analyses assez développées à propos des aspects culturels du mouvement du logiciel libre et de la culture hacker est celle de l'éthique. Ainsi, l'*éthique hacker* est un paradigme sufisamment flexible pour intégrer la multitude de cas qui forment les communautés autour de chaque logiciel. Une démonstration de ceci est que si les hackers et les utilisateurs-développeurs ne s'alignent pas exactement sur les mêmes réalités, la culture hacker est un des points communs à toutes les communautés d'utilisateurs-développeurs.

[8] Rapports de disfontionnement

Pour donner à cette notion la flexibilité désirée, on doit écarter la conception kantienne de l'éthique selon laquelle la loi morale ne vient en aucun cas de l'expérience empirique, mais de ses impératifs catégoriques, pré-établis : "Agis de telle manière que la maxime de ton action puisse devenir le principe d'une législation universelle". Il s'agit pour nous de prendre en compte la *vie sociale de l'éthique* dans le sens des théoriciens sociaux comme Bakhtin [Bakhtin, 1984, 1993]. Ainsi, les règles de vie et de connivence se construisent de manière interactive et dynamique, à partir des pratiques sociales et des expériences vécues dans le cadre communautaire. De cette manière, l'éthique peut se maintenir seulement si on peut observer une relative stabilité des pratiques sociales et une communauté de sens autour des expériences vécues. L'analyse se concentre, alors, sur les évènements qui font les transformations et ainsi, on souligne l'importance d'une auto-modélation communautaire, c'est à dire, du rôle décisif du contexte social, historique et économique.

Dans cette perspective, l'idée reste simple et déjà développée par de nombreux théoriciens [Galison, 1997, Good, 1994, Guterson, 1996], car il s'agit d'exclure les excès de particularismes comme ceux d'universalismes et de rester concentrés à l'observation de faits concrets. Ainsi, les individus adoptent des valeurs et réalisent des choix moraux à travers des actions qui sont influencées par des institutions ou des technologies. C'est cela que nous pouvons apprendre des mots de Coleman et Hill : "Les expériences sociales qui viennent en faveur ou contre une pratique sociale, modèlent la nature des relations individuelles et déterminent les orientations éthiques d'une communauté" [Coleman et Hill, 2005]. En d'autres termes, l'utilisation et l'organisation de cette éthique interagissent avec la communauté ou le groupe à des échelles économiques, politiques et sociales. Au niveau économique, il peut s'agir de production d'un bien, d'un modèle de développement ou d'organisation. Au niveau politique, cela peut être un agenda politique et des manifestations publiques. Enfin, au niveau social, un réseau et des noeuds d'individus et groupes qui s'étendent et traduisent leurs normes au contact de nouveaux domaines et groupes. Pour nous permettre de faire le lien entre modèle de développement participatif, quelques crimes électroniques, les conférences internationales sur l'open-source, Internet, etc, l'*éthique hacker* est une notion heuristique à la compréhension du phénomène du logiciel libre.

2.1.3 La notion d'éthique hacker

On considère, dans la littérature universitaire, que l'expression de *éthique hacker* est utilisée la première fois par le journaliste Steven Levy, dans une des premières oeuvres reconnues sur les cyber-communautés : *Hackers, heroes of the computer revolution* [Levy, 1984]. Á la différence des définitions

utilisées rapidemment après, Levy n'hésite pas a donner de l'emphase à son contenu. Selon lui, l'éthique hacker est un type de prédecesseur moral au logiciel libre et aux communautés open-source. En accord avec cette définition, l'accès à l'information par le hack est totale et tout accès fait découvrir une partie du monde et pour cela se doit d'être absolu, sans limites. À cette fin, l'information et sa circulation doivent être libérées des contraintes institutionnelles en défiant les autorités établies et en restant méfiant par rapport aux légitimités externes à la communauté. Dans ce sens, il est nécessaire de privilégier les formes décentralisées d'organisation et, dans ce contexte, les hackers doivent être appréciés en fonction de leurs qualités techniques et créatives et non pour n'importe quel critère social (race, âge, sexe). Enfin, il y a l'idée que le renforcement technique et pratique des technologies informatiques contribuera à un monde meilleur, une meilleure organisation de la société, plus répartie, juste, démocratique.

À propos des "vrais hackers" que Steven Levy voulait identifier, nous trouvons : John MacCarthy, Bill Cosper, Richard Greenblatt, Richard Stallman. Ensuite, l'auteur décrit une "deuxième génération" de hackers, les *hardware hackers*. Il s'agit des grands noms de l'informatique personnelle, dont une partie participe aux dèbuts de la Silicon Valley : Steve Wozniack, Steve Jobs, Bill Gates, Bob Marsh, Fred Moore. Enfin, une troisième génération apparaît à l'époque des premiers jeux interactifs, les *game hackers* (John Harris, Ken Williams, entre autres).

Néanmoins, en se concentrant sur les externalités gagnantes des communautés informatiques (logiciel, matériel, jeu), l'auteur de *Hacker* semble passer à côté du phénomène social des groupements de hackers comme un tout. De fait, en 1984 – date de la publication du livre – les pratiques hackers dépassaient largement les cadres industriels et contribuaient à la publication d'une immense quantité d'informations qui, à leurs niveaux respectifs, allaient permettre de participer/profiter de ce progrès technique, de l'ère de l'information et des réseaux mondiaux. Dans le cas du logiciel libre, attribuer à Richard Stallman ou à Linus Torvalds la paternité du système GNU/Linux n'est pas seulement une erreur politique, d'oubli des masses d'ingénieurs qui ont contribué aux projets, mais écarte aussi l'observateur ou le lecteur des intérêts, liens et valeurs qui ont motivé des milliers de hackers du monde *online* à travailler ensemble. Ceci nécessite d'accepter comme matériel d'analyse et de témoignage de cette époque les listes d'emails, les forums, les wikis, les manifestes, les débats, les flux d'inforamtions qui ont fait la culture hacker et structuré son éthique.

De cette manière, le philosophe Pekka Himanem a développé une notion d'éthique hacker sensiblement différente. En faisant référence aux travaux de Weber sur l'éthique protestante et le capitalisme moderne, il oppose les va-

leurs hackers aux valeurs protestantes. Selon lui, l'éthique hacker se rapproche plus des définitions de l'*activité* comme elles se trouvent dans les oeuvres de Platon ou Aristote : un travail d'excellence autour d'une recherche de passion, de bonheur et de créativité [Himanem, 2001].

Les analyses de Levy, comme celles d'Himanem, cependant, semblent se cristalliser autour d'un contre-poid au développement péjoratif à l'image du pirate informatique et, dans ce sens, on trouve des travaux encore plus explicites, essayant de "sauver" une réputation ternie [Best, 2003, Hannemyr, 1999]. De manière générale, les conceptions du hacker et du hacking auxquelles ces travaux tentent de s'opposer sont, d'un côté, celles d'adolescents obsédés par Internet et par l'obtention de savoirs dangereux et/ou interdits [Borsook, 2000, Slatalla and Quittner, 1995], et d'un autre côté, des tournois audacieux d'intrusions dans des systèmes privés [Schwartau, 2000]. De fait, il semble que les études jusqu'alors citées s'en tiennent à interagir avec les présupposés existant à propos de la figure du hacker et de son rôle dans la société d'information, sans jamais se demander ce qui fait la particularité de ce groupe en soi. Dans ce sens, l'anthorpologue Gabriella Coleman, dont les travaux font référence tout au long de notre étude, affirme : "La littérature sur les hackers, donc, a tendance à enfermer l'acte de hacker dans un binaire moral au sein duquel les hackers sont soit encesés, soit rabaissés. Cette tendance menace d'isoler plus que d'éclaircir la signification culturelle de hacking informatique"[9] [Coleman and Golub, 2008, p.256].

2.2 Les significations culturelles du hacking

> Ramenée à sa plus simple expression - et abstraction faite de son contenu – l'éthique hacker a son équivalent dans la formule "l'art pour l'art". L'important içi est de saisir que, contrairement à l'activisme politique, l'objet de l'activité hacker, la connaissance et l'exercice de la curiosité, est intérieure à son sujet [Reimens, 2002].

2.2.1 L'agnosticisme politique du mouvement du logiciel libre et de la figure du hacker

Il est évident que de nombreux projets ont été inspirés par la philosophie du Libre. Dans le domaine du journalisme, ont été mis à disposition des

[9]"The literature on hackers, thus, tens to collapse hacking into a moral binary in which hackers are either lauded or denounced. This tendency threatens to obscure more than it reveals about cultural significance of computer hacking."

archives entières (ex : BBC) et s'est développée la figure d'un lecteur participatif – le lecteur-rédacteur – qui construit et discute les flux informatifs, particulièrement dans les tendances du web 2.0. Dans le domaine de la loi, on trouve de nouvelles formes contractuelles pour réguler les productions intellectuelles et artistiques inspirées par les licences libres (*Creative Commons*). Dans l'éducation, des cours sont mis en ligne comme c'est le cas, au MIT, de l'*Open Course Ware*, et ceci sur des plate-formes technologiques libres. Ces projets sont tous liés à un contexte de réseau et d'interconnexion caractéristique de l'Internet et de la société d'information. Cependant, comme l'affirme Christopher Kelty, les significations culturelles du logiciel libre vont au-delà du simple diagnostic de la société d'information et prouvent une réorientation plus spécifique, pour autant qu'elle a à voir avec des pratiques techniques et légales détaillées et spécifiques. De même, il s'agit d'une réorientation plus générale car culturelle et pas seulement économique ou légale. Une preuve de ceci est qu'avec Internet, la gouvernance et le contrôle de la création et la dissémination du savoir a changé considérablement, opérant ainsi une "réorientation du pouvoir et du savoir" qui questionnent profondément la pertinence et la légitimité du système de propriété intellectuelle [Kelty, 2008].

Dans ce contexte, bien qu'il soit clair que la vie politique du code libre et ouvert soit déjà avancée du fait de ses activités et influences, la figure du hacker et des actants du mouvement refuse n'importe quelle affiliation politique. C'est ceci que note Gabriella Coleman en disant : "Alors qu'il est parfaitement acceptable et conseillé d'avoir un groupe sur le logiciel libre à une conférence anti-globalisation, les développeurs de logiciel libre recommandent qu'il soit inacceptable de revendiquer que le libre a pour but l'anti-globalisation, ou à ce propos, n'importe quel programme politique."[10] [Coleman, 2004, p.507].

Il semble adéquat d'observer ici que cet *agnosticisme politique* peut s'exprimer à travers diverses défenses contre les tentatives de donner un sens politique à un acte, une production ou une situation. Dans le domaine de la sécurité informatique, le critère principal est la perfection du système de protection. Pour cela les considérations d'ordre idéologique sont écartées, tout comme le fait pour une technologie d'être libre ou non, au profit de critères comme : l'ouverture code, des mises à jour rapides et une certaine stabilité de la technologie. Dans ce sens, on peut citer le commentaire d'un participant du canal *#hack* du serveur IRC Freenode, lors d'un débat sur le logiciel libre : "ce n'est tout à fait "peu importe"...Si les outils sont ouverts, tant mieux ! on travaille tous sur des systèmes ouverts en général [FreeBSD, Linux] ; ce

[10]"While it is perfectly acceptable and encouraged to have a panel on free software at an anti-globalization conference, FOSS developers would suggest that it is unacceptable to claim that FOSS has as one of its goals anti-globalization, or for that matter any political program."

qui est important c'est d'avoir la technologie la plus sûre, la plus configurable et contrôlable. C'est ça qui fait qu'une technologie est bonne pour un boulot de sécurité, pas le fait qu'elle soit libre ou pas."[11]. Ainsi le site *insecure.org*, qui tient à jour une liste des 10 meilleurs outils de sécurité informatique[12], commente à peine le type de licence qui accompagne les logiciels. Les critères privilégiés sont le prix (gratuit ou non), la compatibilité avec les différents systèmes (Linux, windows, OSX, BSDs) et l'accès au code source. Le logiciel qui arrive en tête de liste depuis des années, Nessus, un outil pour identifier des failles de système, est payant et de code fermé.

Dans diverses activités informatiques, parmi lesquelles l'administration réseau, l'analyse de système d'information, la programmation Web, le refus de l'idéologie se manifeste par exemple à l'encontre du chef de projet, dont les capacités techniques sont fréquemment réduites et influencées par les effets de mode. Tout au long de notre travail, nous avons rencontré de nombreux témoignages dans ce sens, particulièrement dans le domaine du développement web, où les programmeurs cohabitent avec des supérieurs tournés vers la communication et le marketing publicitaire. L'idéologie apparaît, alors, comme un moyen pour l'ignorant de la technique, de se familiariser avec les limitations du concepteur. Ainsi, il construit un discours permettant de justifier d'options technologiques, de budgets, de délais, entre autres. De cette manière, on arrive à une situation où le responsable de projets, souvent dans une situation hiérarchique ascendante, réalise des choix à partir d'informations tronquées que les concepteurs vont devoir suivre sans pouvoir se demander si ce sont les plus aptes à garantir le meilleur service. Nous pouvons identifier un tel phénomène dans le témoignage de Marcelo, administrateur réseau dans une grande université publique brésilienne :

> On devait programmer un truc pour la SOFTEX[13]...Là, réunion dans le bureau du coordonnateur du projet acompagné de son étudiant qui "comprend l'informatique". L'idée est claire, simple, il n'y a pas de problème jusqu'au moment où le mec nous dit qu'on doit programmer le truc en Java, parce-que Java c'est le super-langage qui va gagner de tous les autres, parce que il y a la machine virtuelle super-mega-adaptable, blahblahblah...Tous croient aux effets de mode, ça leur donne des airs de spécialistes...alors...Je réponds que Java n'est pas la meilleure solution pour ce projet...alors là commence la machine à propagande

[11] #hack@irc.freenode.net, 10 novembro 2008

[12] http ://sectools.org/

[13] Association pour la promotion de l'excellence du logiciel brésilien (*Associação para Promoção da Excelência do Software Brasileiro* – SOFTEX

> TI [Technologie de l'Information], et pour lui donner un appui, l'étudiant-qui-comprend-l'informatique dis oui à tout : Java est la meilleure solution dans 100% des cas ! . . .alors je défie le petit étudiant de me dire si c'est possible de programmer telle chose en Java. . .ils pensent, tentent de changer de sujet, etc, etc, et alors, conclusion brillante du gros gars : Si c'est pas 100%, c'est 99% ! . . .[14]

En racontant cet évènement, l'informateur donne un exemple de situation où l'idéologie ("la propagande TI") vient faire préjudice à son travail, et l'oblige de réaliser une tache avec des options technologiques non-optimales, cela pour des raisons qui paraissent "absurdes" ; c'est à dire, des "croyances" de quelqu'un avec une position hiérarchique supérieure et une connaissance technique incomplète et influencée par les effets de mode. Ici, le "mec de SOFTEX" doit probablement baser son choix du langage de programmation Java sur les tendances internes de son institution de promotion du logiciel brésilien ("Java est vu comme le meilleur langage, alors un programme écrit en Java est meilleur") se désassociant ainsi du raisonnement logique tourné vers l'efficacité de la technologie.

Bien que cet exemple illustre une polémique technique qui ne puisse pas être directement qualifiée de "politique", il semble que c'est la même logique qui détermine la conscience politique de ces actants. Ainsi la figure du responsable apparaît comme liée aux logiques de ses propres intérêts, ou du moins non à ceux des concepteurs. De l'autre côté, l'exigence d'excellence technique crée une rationalité qui protège les concepteurs du politique. Dans ce sens, le mouvement du logiciel libre ne prétend pas gagner les faveurs d'une politique publique ou rendre favorable un décision, mais continuer à évoluer en toute autonomie, pour se dédier au meilleur développement technologique possible.

Ces observations permettent de mettre en relief le fait que les littératures anarchiste et libérale font référence au mouvement du logiciel libre comme exemplaire de leurs modes respectifs de production et d'organisation. D'un

[14] Devíamos programar o troço para a SOFTEX. . .Ai reunião no escritório com o coordenador do projeto acompanhado de seu estudante que "manja de informática". A idéia é clara, simples, não tem problema até o cara falar para gente que deve programar o negócio em Java porque Java é a super linguagem que vai ganhar de todos os outros, porque tem a 'máquina virtual' super mega adaptável, blábláblá. . .Todos acreditam na moda, dá a eles ares de experto. . .Então. . .Eu respondo que Java não é a melhor solução para esse projeto. . .E ai começa a máquina a propaganda TI, e para dar um suporte, o estudantezinho "que manja de informática" apoia todas, Java funciona em 100% dos casos. . .Ai, desafio o estudante de me dizer se é possível de programar tal coisa em Java. . .Eles pensam, tentam mudar de assunto, etc, etc e ai conclusão brilhante do gordo : então se não for 100%, é 99%. . .

côté, les analyses libérales trouvent ici un exemple concret de "main invisible", d'absence d'intervention de l'État ou de n'importe quelle autre autorité externe aux processus de décisions internes. Les auteurs développent néanmoins une conscience propre de leur environnement, par un processus libre, et parviennent à prendre des décisions, pour faire des choix tournés vers leurs intérêts qui, joints, créent et maintiennent la communauté du logiciel libre. D'un autre côté, on trouve des auteurs qui voient ici la réalisation concrète de formes anarchistes d'organisation [Moglen, 1999, Gross, 2007]. On souligne alors la haute productivité réelle de l'autogestion des travailleurs, et les racines logiques du défi à l'autorité. De cette manière, en créant autant une alternative possible au néolibéralisme et à l'anarchisme, qu'une reformulation de ceux-ci, l'agnosticisme politique du mouvement du logiciel libre et de la figure du hacker protège son exigence d'excellence technique de directions politiques prédéterminées.

2.2.2 L'information comme passion et la typologie des politiques informelles

En essayant de trouver l'essence du hacking, c'est à dire, un point fixe d'analyse raisonnablement indépendant des pré-supposés du sens commun, l'anthropologue Gabriella Coleman va se concentrer sur la relation que maintiennent les hackers avec l'information. Ainsi, un point commun entre tout les utilisateurs-développeurs, hackers et nerds, est l'amour donné à la liberté de l'information et à sa libre circulation [Coleman, 2003]. Il s'agit d'un sentiment très fort, maniaque et esthétique, proche de *l'amour fou* décrit dans les oeuvres de l'anarchisme ontologique, comme un moyen de se réaliser entièrement comme individu et de défier les structures de la société [Bey, 2004, 1991]. Selon les mots de Bruce Sterling, les hackers sont "possédés non seulement par la curiosité, mais par une véritable luxure du savoir" [15]. Dans une recherche de l'origine d'une telle passion, Gabriella Coleman partage son expérience :

> Mon expérience avec le logiciel libre défend ce principe. L'esprit d'exploration qui forme les bases du hacking doit commencer par démonter la mixture familiale, au grand dam de la mère : ensuite amène à apprendre à programmer à 5 ans, pour la joie de l'union parentale, ensuite se transforme en s'enfermer dans sa chambre pour lire tout les manuels d'informatique, lesquels sont confondus par les parents avec de l'inquiétude pré-adolescente ; ensuite, il s'agit d'apprendre toutes les entrées et sorties de ce système

[15] Cité dans [Coleman, 2003]

> d'exploitation appelé Unix, en découvrant tout les traits typographiques et temporels de l'Internet, au grand étonnement de l'anthropologue ; et, enfin, passer son temps à contribuer, en écrivant du code pour des projets ouverts, souvent encore, en dépit de la consternation des parents[16][Coleman, 2003, p.298].

Après cette discussion, il semble opportun de faire référence à notre expérience au sein d'un département d'administration de réseaux et systèmes d'une université publique brésilienne, comme à celle avec de nombreux professionels et utilisateurs-développeurs de divers domaines de l'informatique. Cela permet d'affirmer que l'élément de passion pour l'information est déterminant dans la différenciation entre "l'armée de réserve" des professionnels des TI, intéressés par l'acquisition d'un savoir et de sa valeur directe sur le marché du travail, et ceux qui, en plus de ces préoccupations matérielles, développent un intérêt constant pour l'appropriation, la production et la discussion de l'information. Au sein de ce département d'informatique, la division des tâches semblait plus déterminée par la capacité de chacun à "s'amuser" face à un problème, que par la formation ou le rang. Le terme "s'amuser", utilisé par les propres acteurs, se réfère principalement aux aptitudes des employés à se responsabiliser, c'est à dire, à percevoir les ramifications d'un problème, se confronter à d'autres erreurs possibles, de tenter les résoudre de la manière la plus systématique et de s'étonner, c'est à dire, s'amuser avec les paradigmes utilisés dans ce processus et les croiser avec des considérations techniques et théoriques afin d'améliorer leur conception générale du système. De cette manière, la curiosité et l'intérêt personnel pour le fonctionnement des technologies utilisées paraissent être le critère déterminant pour juger de la qualité d'un employé. Un informateur illustre :

> chaque fois qu'un nouveau stagiaire entre [dans le département d'informatique] je lui dis : l'Université ne va pas t'apprendre un travail mais te donner une formation générale... Tu va apprendre les choses en mettant les mains dans la pâte, avec des problèmes techniques, concrets, en cherchant des solutions, des petits trucs simples et efficaces. Plus tu cherches des raisons derrière un problème, genre : les structures du système, comment marche le ré-

[16]"My experience with free software support this fundamental tenet. The spirit of exploration that forms the basis of hacking might start by taking apart a household blender, much to a mother's horror : then lead to learning how to programme at the age of 5, much to the delight of the parental unit ; then transform into locking oneself in the bedroom to read every computer manual, wich parents duly confuse with pré-teen angst ; then learning every last topographical and temporal feature of the Net much to the amazment of the anthropologist ; and finally to volunteering their time to code on fre software projects, often to the dismay, again, of their parents."

> seau, le protocole, comment dialoguent deux ordinateurs, plus tu vas créer des réparations expertes, intelligentes, qui vont rester et par-dessus lesquelles on va pouvoir travailler et développer de nouvelles choses. Ça, il y'en a pas beaucoup qui comprennent, hein ? lors u dernier recrutement de stagiaires, il y en avait plus de 15 dans mon bureau et aucun d'entre eux n'a su me dire ce qu'est une base de données... tu le crois ?

Comprendre cette caractéristique du hacker, qu'il soit technicien de laboratoire ou utilisateur-développeur dans un projet libre, permet d'observer que ce qui défini le plus cette figure c'est sa recherche "absurde" pour une information libre, abondante et de précision et que c'est à partir de cela que se construit l'interaction de l'actant avec son contexte social. L'utilisateur-développeur veut une information libre, capable de circuler et d'être modifiée, mais cela se réalise dans un environnement de micro-restrictions qui limitent son activité. Il s'agit de restrictions légales, culturelles, politiques, économiques, qui perturbent une activité subjective d'auto-réalisation. Á propos du logiciel libre, les restrictions les plus évidentes sont légales, c'est à dire, au sein du code, celui que l'on peut ouvrir et modifier, et l'autre qu'on ne peut qu'exécuter. Parmi d'autres options, la retro-ingénierie permet d'obtenir une partie d'un code fermé, mais cela enfreint les droits d'auteurs. De même certaines innovations libres sont utilisées et fermées par des technologies propriétaires. Cela illustre en partie le fait que les frontières entre logiciel libre et propriétaire ne sont pas si claires que ses deux acteurs le laissent entendre. ne plus du fait qu'il y ait de nombreux outils et de protocoles de piratage sous licence libre (ex : Bittorrent), une des origines du mouvement du logiciel libre est d'une certaine manière le *hacking* du système Unix de AT&T (cf. 1.2.2). Cette confusion peut être explorée par les propres compagnies de logiciel propriétaires comme l'observe Túlio Vianna à propos des rapports états-uniens sur le piratage. Par exemple, la *United State Trade Representative* (USTR), associe les parts de marché du logiciel libre à celles du piratage pour dénoncer les crimes contre les droits d'auteurs [Vianna, 2006]. Cependant, le contexte de restrictions auquel se confronte le *geek* dans ses explorations va bien au-delà des considérations légales qui sont les plus médiatiques. Ainsi on trouve divers génériques d'interaction avec la société et ses structures et dans ce sens, Gabriella Coleman a identifié trois idéaux-types de logiques propres à des éthiques différentes de l'information :

Les politiques de transgression : le *undergound hacking*

La partie transgressive du monde du *hacking* est son visage le plus fantasmé et médiatique, mais aussi le plus déprécié. Dans tous les cas, elle

reste quantitativement, en terme de production de code, de participants et d'institutions, minoritaire. Le *hacking undergound* s'est illustré notamment dans la formation des notions d'ingenierie sociale et de *human data* (données humaines) où la systématisation des comportements humains et de leurs failles, s'exprime à travers un cynisme explicite [Mitnick and Simon, 2002]. On trouve alors une forte critique du libéralisme associée à la notion nietzschéenne de pouvoir et de plaisir. Cependant, "comme la tentative de Nietzsche d'élever le pouvoir créatif de l'individu ne s'est jamais vraiment libérée des notions libérales des Lumières, la pratique du *hacking underground* représente plus une radicalisation des fondements du libéralisme, qu'une véritable rupture"[17] [Coleman and Golub, 2008, p.263]. Cette éthique de la transgression est une critique politique, qui se manifeste notamment par le culte du "plaisir d'être surveillé" et de "l'interface entre la surveillance et son échappatoire" [Hebdige]. L'information est "bonne", "agréable", si elle est interdite et si son aquisition nécessite une transgression.

Les politiques de technologies : la cryptoliberté

La cryptographie est l'encodage de données par un algorithme réversible au moyen d'une clé de données secondaires (décodage). Cette technologie est utilisée à des fins de confidentialité, d'authenticité et de contrôle d'intégrité. Dans le domaine de l'informatique, la première clé publique a été publiée en 1975 par Whitefiels Diffie et Martin Hellman (MIT). L'usage de la cryptographie s'étend alors aux institutions et aux entreprises, mais il n'existe pas encore de solutions pour les ordinateurs personnels.

En 1991, alors que le Sénat américain s'apprêtait à voter une loi pour interdire l'utilisation privée de la cryptographie, Phil Zimmerman publie la première clé publique utilisable par des particuliers (PGP - *Pretty Good Privacy*), se rendant ainsi coupable de désobéissance civile et encourant une accusation de trahison. Á cette occasion, l'auteur du programme développe tout un discours pour se défendre sur la scène médiatique :

> Si la privacité est hors-la-loi, seuls les criminels auront le droit à la privacité : [...] PGP permet aux gens de se charger de leur privacité. Il y a une nécessité sociale pour cela et c'est pourquoi je l'ai écrit.
>
> Phil Zimmerman

[17] "[...]just as Nietzsche's attempt to elevate the creative powers of the individual never fully succeeded in definitively escaping the orbit of the Enlightenment's liberal notions, so too, the practice of the hacker under-ground represents merely a radicalization, rather than a complete break from, the moral claims of liberalism."

Ceci est le début de la formation de l'éthique crytpo-libertaire, renforcée par la création en 1992 des Cypherpunks, association de hackers militant pour les droits civils. Ils travaillent sur des technologies de privacité individuelle et militent contre les lois la limitant. Sur le plan politique, ce sont des valeurs libérales qui soutiennent ce mouvement, notamment celles de l'autonomie de l'individu et de sa liberté face au gouvernement. Se crée ainsi l'espoir que les technologies puissent résoudre les problèmes sociaux, car elles reformulent dans le nouveau langage technologique la répulsion libérale à l'intervention de l'état. Concrètement, ce mouvement rassemble des anarcho-capitalistes radicaux, des démocrates, des républicains. Pour certains d'entre eux, il n'y a rien de nouveau dans ce mouvement, il s'agit simplement de la continuation, dans la société de l'information, de la lutte pour les droits constitutionnels.

De plus, le domaine de la cryptographie est un bon exemple de l'influence des communautés libres sur les structures d'une société, en rendant difficile l'interdiction de l'usage d'une technologie, en la mettant libremment à disposition. Ainsi, dans un pays comme la France, où l'usage de la cryptographie était interdit, les nécessités provoquées par les réalisations concrêtes des communautés (libre utilisation, carence sociale) ont largement contribué à l'autorisation de son usage en 1999.

Les politiques d'inversion : le mouvement du logiciel libre

Parallèlement au mouvement cryptolibertaire, d'autres hackers vont développer une autre vision de la sécurité. Pour Richard Stallman, fondateur de la FSF, la connaissance ne doit pas faire l'objet d'orientation, car le bénéfice tiré d'une information est toujours fait au détriment de la communauté. Stallman militait au sein des bureaux du MIT en laissant sa machine sans aucun mot de passe pour que ses fichiers soient mis à disposition de tous avec un message d'accueil expliquant sa philosophie de l'information. Lorsque la FSF a été créée en 1984, s'est développée une pédagogie qui a ouvert le monde des hackers à l'extérieur. Leurs actions respectaient la loi et ils se servaient d'elle pour se protéger. La création de la Licence Publique Générale (GPL) délimite une zone légale de sécurité, de publicité, où les codes restent ouverts. C'est une notion de liberté positive qui est mise en avant, de liberté par l'ouverture, et non une liberté par la négative, par la fermeture ou le secret, comme cela peut être le cas avec la cryptographie.

Cette idée d'inversion se trouve aussi dans l'article *Beating them at their own game* (les battre à leur propre jeu) [Best, 2003] où Kirsty Best démontre que le Libre représente pour ses utilisateurs-développeurs, plus un investissement qu'un rejet des structures sociales existantes autour des nouvelles technologies. Il ne s 'agit pas de promouvoir un changement radical, mais

"d'entrer dans le jeu" des structures capitalistes pour redéfinir ses termes à propos des technologies de l'information et des nouveaux moyens de communication.

Les politiques de collaboration : le mouvement *open-source*

Dans notre étude, ce type-idéal a été ajouté à la typologie de Gabriella Coleman, dans le but de marquer la différence entre l'éthique *open-source* et celle du Libre, nous permettant ainsi de souligner la convergence de plusieurs éthique dans celle étudiée ici.

Associés par "coincidence" [Torvalds and Diamond, 2001] au projet GNU, Linux et son projet technologique vont favoriser la variation "ouverte" du mouvement du Libre, alors appelé à se radicaliser. Le logiciel ouvert, met en avant un modèle de développement non seulement bon, mais surtout efficace. La liberté de l'information libère un *entertainment*, un mérite, qui sont des motivations considérées beaucoup plus efficaces qu'un simple salaire pour inciter à la participation à un espace collaboratif productif. Cette éthique hacker est très développée dans l'oeuvre de Himanem, préfacée par le créateur de Linux, où est soulignée la flexibilisation d'un travail entendu comme hobby et passion, qui peut ne pas recevoir une rémunération directe systématique [Himanem, 2001]. L'idéologie ainsi transmise est plus économique que politique. La liberté de l'information est revendiquée parce que nécessaire et stratégique, néanmoins il faut aussi pouvoir en tirer un bénéfice et favoriser ainsi la création de valeurs.

Dans cette nouvelle économie collaborative, la "wikinomia"[18], "la capacité de rassembler les talents individuels et les entreprises dispersées est en train de devenir "la" compétence du dirigeant et de l'entreprise" [Tapscott and Williams, 2008] et, ainsi, l'objet du propre *hack*. Dans le domaine du logiciel, ce groupe s'est illustré dans la construction du Web 2.0, agglomérats d'outils participatifs où l'utilisateur donne du contenu à la plate-forme, comme c'est le cas de l'encyclopédie Wikipédia basée sur la technologie Wiki, logiciel libre développé par Ward Cunningham en 1995. De plus, l'éthique à dominante collaborative s'illustre dans la sphère technologique en créant des algorithmes performants visant à créer des statistiques dédiées à l'analyse des comportements des utilisateurs d'Internet. Il s'agit alors de "programmer l'intelligence collective"[19], O'Reilly, 2007. et de traiter les informations

[18]Wikinomia est le nom donné à l'économie collaborative à partir du nom de la plate-forme collaborative Wiki qui, par exemple, sert de base à l'encyclopédie en ligne Wikpédia.

[19]Titre d'un livre récent ayant fait référence dans le domaine de la programmation du web collaboratif : SEGERAN, Toby, *Programming Collective Intelligence :* Bulding Smart Web 2.0 applications

accumulées par l'observation des utilisateurs, pour, par exemple, suggérer des "produits assimilés", "un partenaire idéal", etc. Si la programmation de la collaboration développe sa propre éthique de l'information, ses outils technologiques s'inspirent beaucoup des mouvements vus précédemment : elle associe l'ingénierie sociale du *hacking underground* et est viscéralement liée au libre accès aux informations, afin de pouvoir les traiter, et à leur stricte confidentialité, pour garantir la privacité des utilisateurs et empêcher les contestations judiciaires d'atteinte à la vie privée.

*

En observant cette typologie, on note une *heteroglossia* [Bakhtin, 1984, 1993], c'est à dire une diversité au sein du même code linguistique, dans lequel on trouve une discussion incessante sur la liberté. Ce qui constitue le discours moral des hackers et ce qui différencie leurs éthiques, c'est l'élaboration d'un sens autour de ce qu'est la liberté et de ce que signifie être libre. Cette diversité donne un dynamisme aux communautés hackers, qui passent alors d'une variété de discours à une autre et changent ainsi de répertoires de références sans beaucoup se préoccuper des contradictions de contenu, du style ou des effets politiques. De plus, si la trajectoire discursive qui réalise la collaboration dans les communautés est en constante négociation, au fil du temps, des points fixes apparaissent. La défense de la liberté d'information cohabite avec une tradition libérale qui trouve alors une nouvelle visibilité et un nouveau discours hétéroclite en harmonie avec l'ère digitale.

Néanmoins, on peut affirmer que ce qui unit ces nouveaux discours c'est une pratique partagée de la programmation. En effet, quel que soit le domaine technologique ou politique de ces idéaux-types, ils désignent des communautés qui ont pour ressemblance, pour activité commune, le fait d'écrire du code. Dans ce sens, Coleman affirme que "la liberté du logiciel libre, bien qu'influencée par des sensibilités libérales majeures, est fondamentalement modelée par les pragmatiques de la programmation et le contexte social de l'utilisation de l'Internet"[20] [Coleman, 2004, p.509].

[20]"The freedom of free software, while influenced by wider liberal sensibilities, is fundamentally shaped by pragmatics of programming and the social context of internet use."

2.3 Les pragmatiques de la programmation comme outil pour aborder et suivre l'interaction des utilisateurs-développeurs avec l'information

Dans un effort de définition de l'acte de programmer, Andrew Goffey utilise la définition de "dire" faite par Michel Foucault dans *L'archéologie du savoir* : "~~Dire~~ [programmer], c'est faire quelque chose – quelque chose de différent que d'exprimer ce que quelqu'un pense, traduire ce que quelqu'un sait, et quelque chose de différent que de s'amuser avec la structure du langage"[21] [Goffey, p.14]. En observant les utilisateurs développeurs comme programmeurs, une dualité semble diriger leur tâche : d'un côté, il s'agit d'un acte très précis, d'un traitement rationnel d'une situation rationnelle dans un contexte rationnel ; cependant, d'un autre côté, le discours entre les programmeurs sur leur objet, l'ontologie que les auteurs développent sur leur oeuvre, appellent de nombreuses références subjectives, intersubjectives, et proches de l'expression d'une nature artistique et esthétique du code (2.3.1). En observant et analysant ces natures artistiques et normatives de l'acte de programmer, nous pourrons, alors, exposer notre paradigme d'analyse, c'est à dire, les pragmatiques de la programmation comme vecteur de l'agnosticisme du mouvement du Libre et de sa relation à l'information.

2.3.1 La programmation comme art et régulation

La programmation comme art et l'esthétique du code

"Poète à poète. Je t'imagine
en marge du langage, au début de l'été
à Wolfeboro, New Hampshire, écrivant du code.
Tu n'as pas la notion du temps. Ni celle des minutes.
Elles ne peuvent pas atteindre ton monde,
ton ordinateur gris
avec quand déjà maintenant jamais et une fois.
Tu avais perdu les septs autres.
Celui-ci est le huitième jour de la création.
...

[21] "To ~~speak~~ [program] is to do something – something other than to express what one thinks, to translate what one knows, and something other than to play with the stucture of language."

J'écris sur un écran bleu
comme n'importe quelle montagne, comme n'importe quel lac,
composant ceci
pour te montrer comment le monde recommence :
Un mot à la fois.
D'une femme à une autre."[22]
Code, Boland, 2001.

L'art de la programmation est le titre d'un ensemble de quatre volumes reconnu comme une des plus importantes oeuvres didactiques du domaine de l'informatique, et nommé comme l'une des douze majeures monographies scientifiques par le journal *American Scientist*. La rédaction de ce livre a commencé en 1962 lorsque la maison d'édition Addison Wesley proposa à Donald E. Knuth, alors docteur en mathématiques de 24 ans et candidat à un poste de professeur, d'écrire un livre sur les compilateurs. Jusqu'à aujourd'hui, le livre est actualisé fréquemment par l'auteur et un cinquième volume est en préparation pour 2015. Selon Maurice J. Black, cette oeuvre suit un courant d'enseignement et de commentaires des programmes et de la programmation comparable à un effort de critique littéraire. Avant *L'art de la programmation*, les *Commmentaires de Lions sur le sixième version de Unix* de John Lions accompagnaient déjà les programmeurs dans leur étude du code source du système de AT&T, alors mis à disposition. Plus qu'une simple étude, il s'agissait en fait de la lecture d'une entité esthétique dont les commentaires révélaient les sens, particularités et logiques [Black, 2002].

Au-delà du fait que les écrits sur le code puissent être considérés comme une critique littéraire, le code lui-même apparaît comme une oeuvre en soi. Ce point de vue est parfaitement incarné par les théories de la programmation littéraire (*literate programming*) développées par Donald Knuth :

> Un adepte de la programmation littéraire peut être vu comme un essayiste, dont la principale préoccupation est d'exposer avec excellence de style. Un tel auteur, son thesaurus en main, choisit les noms de ses variables avec attention et explique ce que chacune d'elles signifie. Lui ou elle s'efforce de faire un programme qui soit compréhensible, car ses concepts ont étés introduits dans un ordre

[22]Hommage de la poètesse Eavan Boland à Mme Grace Murray Hopper (1906 – 1988), programmeuse d'un compilateur pour le language COBOL, pendant les années 1940.
"Poet to poet. I imagine you / at the edge of language, at the start of summer / in Wolfeboro, New Hampshire, Writing code. / You have no sense of time. No sense of minutes, even. / They cannot reach inside your world, / your grey workstation / with when yet now never and once. / You have missed the other seven. / This is the eighth day of creation.[...] I am writing at a screen as blue /as any hill, as any lake, composing this / to show you how the world begins again : / One word at a time. / One woman to another."

> qui est le plus adéquat à l'entendement humain, en utilisant un mélange de méthodes formelles et informelles qui se renforcent les unes les autres.[23] [Knuth, 1983, p.1].

De cette manière, l'analogie faite entre la programmation et la littérature reformule une méthodologie de l'acte d'écrire du code. "Selon Knuth, le meilleur code n'est pas celui écrit depuis la perspective pragmatique d'un ingénieur, mais celui écrit depuis la perspective artistique d'un auteur. L'économie du style, la clarté de l'expression et l'élégance formelle sont aussi essentielles à la bonne programmation qu'elles le sont à la bonne écriture.[24] [Black, 2002]. Plus radical encore dans le sens de considérer l'acte de codifier comme une pratique poétique, le mouvement de la Poésie Perl (*Perl Poetry*) utilise le langage de programmation Perl, caractérisé par l'usage de fonctions désignées en termes anglais "naturels", pour écrire et traduire des poèmes compilables (Figure 2.1, p.43).

FIG. 2.1 – The Coming of Wisdom with Time (1910, 2000)

Though leaves are many,
the root is one ;
Through all the lying
days of my youth
I swayed my leaves and
flowers in the sun ;
Now I may wither into the
truth

William Butler Yeats, 1910.

```
while ($leaves > 1) {
        $root = 1;
}
foreach($lyingdays{'myyouth'}) {
        sway($leaves, $flowers);
}
while ($i > $truth) {
        $i--;
}
sub sway {
        my ($leaves, $flowers) = @_;
        die unless $^O =~ /sun/i;
}
```

Wayne Meyers, 2000.

```
perl The\ Coming\ of\ Wisdom\ with\ Time
Died at The Coming of Wisdom with Time line 12.
```

La métaphore de l'art n'est pas étrangère aux faits de langages communs

[23] "The practitioner of literate programming can be regarded as an essayist, whose main concern is with exposition and excellence of style. Such an author, with thesaurus in hand, chooses the names of variables carefully and explains what each variable means. He or she strives for a program that is comprehensible because its concepts have been introduced in an order that is best for human understanding, using a mixture of formal and informal methods that nicely reinforce each other"

[24] "For Knuth, the best code is written not from the pragmatic perspective of an engineer, but from the artistic perspective of an author. Economy of style, clarity of expression, and formal elegance are as essential to good programming as they are to good writing"

au sein du milieu informatique. Comme le note Samir Chopra et Scott Dexter, la "beauté" du code est un sujet confortable pour les informaticiens et le langage pour la décrire a de nombreux adjectifs émotifs. Une suite d'instructions est belle ou laide, propre ou sale, légère ou pesante, fantastique, impressionnante, horrible, etc [Chopra and Dexter, 2007]. Dans ce sens, un extrait du code source du kernel Linux de 1990 l'illustre (Figure 2.2, p.44) :

FIG. 2.2 – Extrait du Kernel Linux 0.11 (1990)

```
/*
* Yeah, yeah, it's ugly, but I cannot find how to do this correctly
* and this seems to work. If anybody has more info on the real-time
* clock I'd be interested. Most of this was trial and error, and some
* bios-listing reading. Urghh
*/
#define CMOS_READ(addr) ({ \
outb_p(0x80|addr,0x70); \
inb_p(0x71); \
})
(Linux Kernel 0.11 main.c )
[...]
#define outb(value,port) \
__asm__(''outb %%al,%%dx'':''a'' (value),''d'' (port))
#define inb(port) ({ \
unsigned char_v; \
__asm__ volatile (''inb %%dx,%%al'':'' = a'' (_v):''d'' (pot)); \
_v; \
})]
(Linux Kernel 0.11 /include/asm/io.h )
```

Bien qu'il fonctionne correctemment, ce code est décrit comme "laid" (*ugly*) par son auteur, pour ne pas être la meilleure solution au problème rencontré, dans ce cas, synchroniser l'horloge du système opérationnel avec l'horloge physique du hardware. On pourrait croire que si un code fait fonctionner un outil et que son résultat est celui que l'utilisateur attend, le code est valide, correct. Néanmoins, la relation du programmeur avec son code, avec les autres programmeurs et avec l'objet programmé – ici le système opérationnel – exige une élégance logique qui doit faire sens pour ses actants. Ce code doit réguler l'interaction du logiciel avec le matériel, de l'utilisateur avec ces deux derniers et des développeurs à venir avec l'auteur. Ces interactions forment un objet abstrait, sujet à beaucoup d'interprétations et de constructions différentes et, comme l'artiste extrait une oeuvre unique de son corps ou d'une représentation, on peut considérer "le code comme une ten-

tative de capturer la beauté d'un objet abstrait, l'algorithme" [Chopra and Dexter, 2007, p.77]. En comparant ceci avec la citation à suivre d'un critique littéraire tentant définir la poésie : "Donc ! La poésie serait le moment où le mot exact, l'agencement exact, est obtenu pour qu'enfin le réel qui a pu être cerné, soit restitué grâce au matériau !"[25]. Un algorithme, pour être un montage unique qui délimite les interactions et pour être restitué grâce à un programme, est, alors, comme la poésie, un art.

L'objectif de comprendre le code comme un art est de pouvoir délimiter les caractéristiques du logiciel libre propre à sa nature artistique. Puisque l'esthétique nous dit quelque chose sur l'objet logiciel et sa création, sur la relation entre le programmeur et son artéfact, parler de logiciel libre, c'est délimiter les spécificités d'un code ouvert dans ces interactions. Le principal argument des communautés FOSS dans ce sens est le fait que, comme l'artiste est influencé par son contexte de production, par les oeuvres avec lesquelles il interagit et desquelles il s'inspire, le modèle de développement collaboratif permet à une esthétique très forte de se structurer. Comme un auteur doit lire de bons écrits, un bon programmeur doit s'inspirer de bons codes pour réaliser les exigences communautaires d'esthétique. Dans ce sens, et non sans ironie, Bill Gates affirme :

> Le meilleur moyen pour se préparer [à être un bon programmeur] est d'écrire des programmes, et d'étudier les excellents programmes que d'autres personnes ont écrits...Vous devez vouloir lire les codes des autres, puis écrire le vôtre, puis le faire réviser par d'autres encore. Vous devez vouloir être au sein de ce retour incroyable où vous trouvez des gens au niveau international pour vous dire ce que vous faites de mal...Si jamais vous parlez à un grand programmeur, vous verrez qu'il connaît ses outils comme un artiste connaît ses pinceaux ? C'est impressionnant de voir combien les bons programmeurs ont en commun dans la manière qu'ils ont de développer...Quand vous amenez ces gens à observer un très bon morceau de code, vous obtenez une réaction très, très semblable[26] Bill Gates, in cite[p.83]Lammers1989.

[25]Fabrice Luchini en dédicace au Virgin Megastore à Paris, 17/11/2008 : `http://www.dailymotion.com/relevance/search/luchini/video/x7yrj5_fabrice-luchini-en-rencontre-dedica_creation`

[26]"The best way to prepare [to be a good programmer] is to write programs, and to study great programs that other people have written...You've got to be willing to read other people's code, then write your own, then have other people review your code. You've got to want to be in this incredible feedback loop where you get the world-class people to tell you what you're doing wrong...If you ever talk to a great programmer you will find he knows his tools like an artist knows his paintbrushes. It's amazing to see how much great

Ainsi le logiciel libre facilite une interaction sans restriction de codes divers. L'inspiration par d'autres codes, en plus d'être possible, est encouragée par une tradition forte de critiques et commentaires au sein des communautés FOSS. Il y a une créativité de groupe qui caractérise le produit final comme étant le "meilleur" code, car écrit par les meilleurs programmeurs, car formés par ce même meilleur code, formé petit à petit au sein de communautés qui ont de très fortes exigences esthétiques. De plus, au sien des communautés FOSS, il y a l'idée que l'exigence traditionnelle d'ésthétique et d'élégance technique fait "survivre" ce que la littérature classique a perdu. Dans ce sens, Maurice Black commente :

> [La critique littéraire] démontre ses engagements politiques en abandonnant les thèmes de la littérature et de l'esthétique [...] et, plus encore, en démontrant son manque de foi en l'esthétique par sa réification de la laideur au niveau de style ou de choix thématique. Dans l'objectif de transgresser et de déstabiliser les canons littéraires à fins politiques, les théoriciens culturels sont maintenant avec une alacrité toute prête pour traiter des sujets comme la folie, la torture, la douleur, la monstruosité, la pornographie et la maladie. La culture informatique, de son côté, a adopté un modèle traditionnel d'esthétique littéraire comme moyen de changement effectif, en trouvant utilité politique et valeur sociale dans un produit bien fini qui est, à la fois, entièrement utilisable et beau à contempler[27] [Black, 2002, p.20].

La programmation comme architecture et la régulation par le code

Quand dans un établissement privé de restauration on revient de la salle de bains vers la salle principale, comme la place d'alimentation d'un centre commerciale, on trouve presque systématiquement une porte donnant accès aux cuisines, au cellier, ou bien à un lieu de repos pour les employés. Sur la porte, on trouve un message en interdisant l'accès : "accès restreint", "entrée

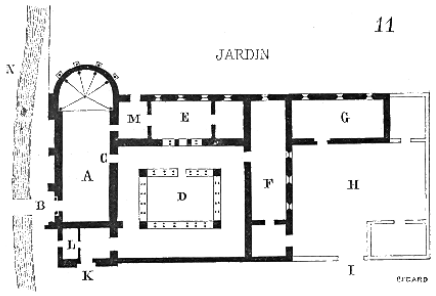
programmers have in common in they way they develepoed...When you get those people to look at great piece of code, you get a very, very common reaction."

[27]"The first has demonstrated its political commitments by all but abandoning literature and aesthetics as its subject matter [...] and even by demonstrating its loss of faith in aesthetics through its reification of ugliness at the level of critical style and choice of subject matter – with the goal of transgressing and destabilizing the literary canon for political effect, cultural theorists now turn with ready alacrity to subjects such as madness, torture, pain, monstrosity, pornography, and disease. Computing culture, on the other hand, has adopted a traditionel model of literary aesthetics as a means of effecting change, finding political utility and social value in the well-crafted product that is at once entirely usable and wholly beautiful to contemplate."

interdite", "réservé aux employés"...Ainsi on se dirige en fonction de son propre statut (client, employé, agent de sécurité) par le chemin qui nous est autorisé. Quand quelqu'un se connecte à son compte Gmail, son ordinateur lit les données d'un serveur d'un centre Google. Le navigateur lit donc des données qui sont physiquement voisines de nombreux autres comptes Gmail. Cependant, une règle l'empêche de diriger la tête de lecture du disque dur vers un autre compte, comme il l'avait fait pour son propre compte. C'est une règle codifiée dans un language-machine qui permet cette opération. Il y a sur ce serveur un système opérationnel qui administre des comptes pourvus d'espaces et de privilèges distincts (utilisateurs différenciés, administrateur de système, etc). Alors, comme dans le centre commercial où une architecture physique vient délimiter des salles et des couloirs dont les accès sont restreints par des normes exprimées par des symboles, le logiciel d'un serveur email vient réguler l'activité des utilisateurs par une architecture et des lois.

FIG. 2.3 – Comparaison entre un plan d'architecture et un code délimitant les espaces disques d'utilisateurs d'un système

Un plan d'architecture



Un code Linux standard délimitant les espaces de divers utilisateurs

```
root:x:0:0:root:/root:/bin/bash
fulano:x:1000:1000:fulano,,,:/home/fulano:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
```

Depuis la publication du livre de Lawrence Lessig, *Code and other laws of cyberspace* [Lessig, 1999], affirmer que le code est une loi qui régule les comportements par coercition est devenu une idée commune au sein des analyses de la société d'information. Grâce à des métaphores détaillées de la loi, de l'architecture, des normes sociales et des marchés financiers, l'auteur montre que le logiciel régule les comportements, que "code est loi". Á partir de cette observation, la réflexion de Lessig va s'articuler selon un syllogisme simple : si les institutions peuvent réguler le logiciel et que le logiciel peut réguler les individus, alors les institutions peuvent réguler les individus par le logiciel. Ainsi, on souligne l'importance de comprendre les implications politiques et

sociales des décisions techniques, car elles incarnent une nouvelle forme de loi. De manière générale aussi, se discutent les changements que les dirigeants doivent promouvoir pour l'Internet.

James Grimmelmann [Grimmelmann, 2005] fait une critique du travail de Lessig et des débats qui naquirent des analogies entre logiciel, loi et architecture. Bien que ces métaphores soient considérées comme heurisitques à la compréhension des rôles sociaux et politiques du code, Grimmelmann souligne le manque de discussion des différences qualitatives entre logiciel et loi et entre logiciel et architecture. Ainsi, si ces analogies donnent beaucoup de sens pour comprendre l'objet logiciel dans son contexte social et politique, plusieurs différences de nature et de mode d'opération différencient ces objets. Pour l'auteur, la comparaison avec l'architecture permet de souligner la nature automatisée et immédiate du logiciel, alors que celle de la loi permet de souligner sa nature plastique. Le logiciel est *automatisé* parce que une fois programmé, il agit sans nécessiter aucune intervention humaine. Au dela de la propre norme, qui une fois établie se reproduit de façon autonome, des algorithmes performants, communément désignés comme "intelligents", peuvent s'adapter aux évolutions des comportements qu'ils régulent. Le logiciel est *immédiat* car, au lieu de s'appuyer sur les bases de la sanction, il empêche simplement l'action de se produire. La coercition n'a pas de degré différent selon son appréhension par les individus, c'est à dire par la peur de la sanction, car l'acte interdit ne peut tout simplement pas etre réalisé. Enfin, le logiciel est *plastique* car il est concevable n'importe quel système qu'un programmeur puisse imaginer et décrire avec précision. Ce dernier point souligne aussi la nature fragile et cassable du code, c'est à dire, que n'importe quel faille qui peut être imaginée, peut être explorée. Dans ce contexte, une caractéristique supplémentaire différencie le code ouvert du code fermé. Un logiciel dont le code est ouvert est transparent, ses modes de régulations sont accessibles à la compréhension des sujets dont il détermine le champ des possibles. D'un autre côté, un logiciel fermé maintient un algorithme caché, comme fait unique et autonome, indépendant de la compréhension que ses sujets en ont.

2.3.2 Les pragmatiques de la programmation comme paradigme d'analyse

Les pragmatiques de la programmation : une interprétation socio-politique d'une notion linguistique

Une des définitions les plus anciennes de la pragmatique est faite par Charles W. Morris : "La pragmatique est cette partie de la sémiotique qui

traite des relations entre les signes et les usagers des signes" [Morris, 1938]. D'un point de vue strictement linguistique, la pragmatique traite avant tout du sens, comme la sémantique, mais pour elle, c'est l'usage qui est fait des formes linguistiques qui détermine le sens qu'elles peuvent avoir. Dans sa branche la plus pragmatique, notamment dans l'oeuvre de Yehosua Bar-Hillel [Bar-Hillel, 1970], on utilise des paradigmes comme l'étude des symboles lexicaux, sens littéral et sens communiqué, actes de langage.

De manière générale, la pragmatique permet de questionner divers faits de langage : quand nous parlons, que faisons-nous ? que disons-nous exactement ? qui communique avec qui ? qui suis-je pour mon interlocuteur ? quel sens est nécessaire à la cohérence de mes propos ? un mot a-t-il un sens littéral ? Quels sont les usages du langage ? [Armengaud, 2007]. Ainsi, parler des "pragmatiques de la programmation" serait questionner de la même manière l'acte d'écrire du code : Que faisons-nous lorsque nous programmons ? Que programmons-nous ? Comment l'utilisateur perçoit-il l'ordinateur ? comment l'ordinateur reçoit-il les instructions des utilisateurs ? Comment se forment les erreurs d'interprétation et de syntaxe ? Et, principalement, dans quelle mesure la réalité du programmeur et sa capacité à programmer interagissent, se déterminent-elles l'une et l'autre ?

Dans un abord plus philosophique de la pragmatique et des interprétations plus politiques et sociales qu'elle permet, la pragmatique a réorienté le regard de la science du langage sur les interlocuteurs. Leurs complexités, mises à nu par son analyse, amènent l'observateur à questionner les concepts de sujet et d'individu, notamment en se concentrant sur l'interlocution. Ainsi, s'il y a une Raison, elle ne peut exister que par la validation intersubjective du savoir. Appliqué à l'esthétique, l'angle d'analyse offert par la pragmatique fait des oeuvres artistiques un ensemble de formes symboliques dont les sens et références dépendent des ses conditions de vie et d'action, de son contexte. L'art est alors une expérience qui a pour conséquence de communiquer avec un contexte qui vient lui donner son sens : "Je ne dis pas que la communication est l'objectif d'un artiste, mais qu'il s'agit d'une conséquence de son travail"[28] [Dewey, 1934].

De cette manière, l'analogie entre la programmation et l'art par l'utilisation de l'idée d'esthétique du code [Chopra and Dexter, 2007] démontre que plusieurs parallèle sont acceptés entre écrire du code et rédiger de la littérature, entre commenter du code et faire de la critique littéraire, entre écrire des scripts comme s'écrit de la poésie [Black, 2002]. Pour cela, en acceptant l'idée que la programmation puisse être considérée comme un des "beaux-

[28]"I do not say that communication is the intent of an artist. But it is a consequence of his work."

arts" [Levy, 1992], le concept de pragmatique aide à écarter les notions de sujet de l'analyse du mouvement du logiciel libre et à se concentrer sur la constitution d'un corpus de code qu'il incarne. Les différentes communautés et leurs interactions autour des projets logiciel constituent une *praxis* qui se révèle à l'observateur par l'analyse des relations développées avec l'information.

La pragmatique de la programmation comme relation avec l'information et ses politiques informelles

L'hypothèse l'usage des pragmatiques de la programmation comme outil d'analyse du mouvement du logiciel libre est que la programmation est un discours sur l'information. Un programme traite, gère, crée et modèle des informations. Les individus qui code ces logiciels créent donc un discours sur ces informations en choisissant la manière par laquelle la machine va les traiter. Chaque protocole de réseau traite les flux d'informations d'une manière différente, et l'effort de configuration fait par l'administrateur d'un réseau déterminé réalise des choix spécifiques qui vont aussi structurer l'information, sa diffusion et sa réception. De ces choix et régulations qui constituent la relation à l'information, les programmeurs construisent une ontologie que le programme vient automatiser. Dans ce sens, la notion de pragmatique permet d'isoler cette tache du programmeur et de développer une vision ample des interactions qui se manifestent autour de l'acte de programmer. Au delà des interactions entre les propres utilisateurs-développeurs, ont peut considérer aussi leurs interactions avec l'objet ordinateur, ses logiques, ses limites. Elles maintienent une relation privilégiée, pour être très normative et décisive, avec une machine qui traite l'information, alors entendu comme *dispositif*, c'est à dire, "un ensemble hétérogène qui résulte du croisement des relations de pouvoir et de savoir" [Agamben, 2007]. Cette notion de "dispositif" est caractéristique de l'effort des communautés FOSS au sein du mouvement technologique. La machine isolée, particulière, se définit, dans ses fonctions et possibles, par les logiciels qu'elle utilise pour traiter les informations que l'utilisateur lui donne. Les normes et structures de tels logiciels résultent des relations de pouvoir et de savoir présentes dans le processus de codification d'un programme. Alors, la pragmatique du programmeur est l'étape la plus fondamentale (après la création du propre langage de programmation) où ces relations se réalisent.

De cette manière, l'information est structurée par l'objet logiciel, et les communautés FOSS produisent un discours sur ces flux d'information qui génèrent un ensemble automatisé et plastique qui donne son contenu au Libre comme mouvement politique et régulateur de l'espace digital. Paradoxale-

ment, comme nous l'avons montré, la pragmatique de la programmation est aussi le vecteur de l'agnosticisme politique du mouvement du Libre du fait que la logique de la programmation détermine et permet l'absence de discours politique apparent, ou, au moins, référençable de manière commune. Néanmoins, c'est un discours spécifique que forment ces pragmatiques, sur la société d'information, qui construit un répertoire de références propre à ses actants. Ainsi, l'utilisateur-développeur crée un discours qui est un acte. Cette idée nous permet de comprendre que le code est à la fois une théorie, une abstraction et une pragmatique. Il est au même moment l'écriture d'un texte, une preuve, et la réalisation d'une expérience concrete. Pour cela, les idées politiques, comme la liberté d'expression, sont implantées par les communautés dans le processus de production et dans l'objet produit. Un exemple est la technologie Wiki, qui, construite au sein de communautés libres avec des dynamiques participatives, permet la création d'un contenu collectif ouvert. La créativité du groupe crée un processus dont les conditions se reproduisent dans le produit [Sawyer, 2003]

En somme, l'effort de l'observateur du mouvement doit être d'émettre des hypothèses au niveau le plus fondamental de ce discours – le code – pour comprendre les constructions des entités politiques du mouvement du logiciel libre au sein de ses politiques informelles (transgression, civisme, inversion, collaboration).

Chapitre 3

Les pragmatiques de la programmation et ses logiques politiques informelles

Nous avons observé dans le chapitre précédent les significations culturelles du mouvement du logiciel libre par le moyen de : a) la notion d'éthique hacker, qui nous a permis de souligner 4 idéaux-types de logiques politiques informelles ; b) la construction de l'idée des pragmatiques de la programmation comme vecteur de l'interaction des utilisateurs-développeurs FOSS avec l'information et ainsi, nous a permis d'émettre des hypotèses au niveau le plus fondamental de l'activité du programmeur – c'est à dire le code – pour comprendre le construction de l'entité politique du mouvement du logiciel libre.

Á cette fin, nous proposons ici une typologie de trois pragmatiques qui se retrouvent dans le discours des communautés libres. En premier lieu, il y a la pragmatique de la *liberté* entendue comme l'idéologie qu'on trouve derrière le mouvement GNU et les premières traces de la culture du logiciel libre. Par exemple, on peut choisir un outil pour le fait qu'il soit sous licence GPL, et ceci comme impératif supérieur à tout autre, afin de créer une solution homogène "libre". De plus, la pragmatique de la liberté est le discours et les interactions du mouvement du logiciel libre qui privilégient les technologies "100% libre", dans le sens où l'entendent les actants.

Au delà, nous trouvons l'idée d'*ouverture* dans les discussions à propos de la comptatibilité des technologies, principalement les protocoles et les formats d'archives. La pragmatique d'ouverture serait, alors, le discours des actants du Libre sur ces problématiques, qui privilégie ces caractéristiques par rapport aux autres.

Enfin, la préocuppation de *sécurité* se rencontre particulièrement dans

les problématiques de stabilité des systèmes d'information et de ses défenses contre d'éventuelles attaques sur le réseau. Ainsi, la pragmatique de sécurité serait reflétée dans les efforts des communautés pour rendre leurs programmes avant tout sûrs, même si cela peut créer des lacunes de performances ou de compatibilité, ou bien même si des licences hybrides protègent alors une partie de leur code.

L'objectif dans ce chapitre est de faire dialoguer la typologie des logiques politiques informelles propres à l'éthique hacker, avec les pragmatiques de la programmation, afin que se révèlent les propriétés politiques d'un acte que nous considérons alors comme fondamental, la programmation. Dans ce sens, la pragmatique de la liberté sera examinée à partir de l'étude de la communauté gNewSense (3.1), celle d'ouverture en observant le projet SAMBA (3.2) ; et celle de sécurité par l'analyse des systèmes BSD (3.3). Enfin, nous présenterons un tableau de comparaison des interactions conceptuelles désirées (3.4).

3.1 Les pragmatiques de liberté : la communauté gNewSense

> Le logiciel non-libre n'est jamais une solution, alors, s'il vous plait, ne pas rationaliser, justifier ou minimiser les conséquences de proposer un logicel non-libre comme solution.[1] (5^{e} principes de la communauté gNewSense).

La pragmatique de la liberté est la première qui apparaît au contact des communautés du Libre. Pour cela, les responsables de ces communautés veulent défendre par la technologie le droit à participer aux évolutions librement, car un code ouvert est la condition d'une telle idée de liberté. Ces prétentions ont été embrassés de manière emblématiques par la communauté gNewSense qui tente de construire une version 100% libre de Linux, car celui-ci n'est pas "libre" dans le sens que tente de promouvoir la FSF et ses communautés proches. La plupart des distributions Linux ont toujours permis l'installation de logiciels restreints, qui, ainsi, constituaient un système sous l'empire de réglementations hétérogênes. Cette banalisation de codes restreints dans l'environement de Linux a fait un pas supplémentaire avec l'adoption en 2006 de morceaux (*blobs*) de codes fermés dans le propre kernel du système. Ces blobs sont alors été incorporés aux systèmes distribués,

[1] "Non-Free Software is never a solution so please do not rationalize, justify, and minimize the consequences of proposing non-free software as a solution".

parmi lesquels un des plus dédiés aux idéaux du logiciel libre[2].

Dans ce contexte, Richard Stallman, fondateur de la FSF et Mark Suttleworth, investisseur de Ubuntu, ont souligné d'une même voix l'importance de maintenir un effort pour construire une version strictement libre du système. Un des participants de la conférence, Paul O'Malley, amène le débat au public par IRC et, avec Brian Brazil, commence à développer une architecture rénovée du système. Les blobs et les répertoires de logiciels fermés et/ou restreints sont retirés et se constitue une communauté autour de ce nouveau système qui reçoit alors le nom de gNewSense (gNS). Les utilisateurs-développeurs se confrontèrent à des difficultés techniques importantes, rendant le système difficile à utiliser pour les nécessités communes et avec des problèmes de compatibilité de matériel. Cependant, actuellement, en Juin 2009, le système est mis à disposition dans sa version 2.2 et réussit à associer toutes les alternatives possibles aux programmes restreints, offrant ainsi une possibilité tierce aux compromis faits par les distributions concurrentes.

Les analyses proposées ici ont été réalisées à partir de matériaux récupérés principalement sur le Forum et les listes de mails de la communauté[3] et des interactions sur le canal IRC[4]. Au-delà des observations des débats sur le vif, et dans les archives de chat, ont été réalisés des entretiens de plusieurs utilisateurs-développeurs, parmi lesquels, Paul O'Malley, un des fondateurs de cette distribution.

3.1.1 Logiques de transgression

Les pragmatiques de la programmation développées par les utilisateurs-développeurs de cette communauté ne sont pas entendues comme transgressive. Le hacking, dans ses aspects *underground*, n'est pas un vecteur de l'effort technologique qui conduit la construction de ce système alternatif. De fait, les utilisateurs-développeurs aiment à se présenter, comme ceux du projet GNU ou proches de la FSF, comme "*gnu hacker*". Pour être un acte profondément politique et civique, les caractéristiques des pragmatiques de la communauté gNewSense doivent être recherchées dans les autres logiques qui constituent notre typologie.

[2]Vote de la communauté Debian en faveur de l'adoption des blobs binaires : 15/10/2006, `http://www.debian.org/vote/2006/vote_007.en.html`

[3]`http://wiki.gnewsense.org/index.php?n=ForumMain.ForumMain`

[4]#gnewsense@irc.freenode.net :6667

3.1.2 Logiques de civisme

Les pragmatiques de civisme apparaissent au sein de gNS comme un ensemble de choix tournés vers un résultat d'intérêt général. De fait, les restrictions techniques qui s'imposent aux "gnu hackers" pour développer le système sur une plate-forme avec des limitations techniques importante, sont un acte positif dans le sens où elle ignorent les alternatives et les compromis réalisés par les autres distributions de Linux. Il y a un refus catégorique des perspectives de développement "réformiste" qui entendent le changement du Libre comme un processus progressif qui nécessite l'utilisation d'outils sous licences restrictives et/ou propriétaires. Cette position des développeurs s'affirme quotidienement sur les plates-formes d'interactions avec les utilisateurs qui demandent de l'aide pour pouvoir utiliser le système et souvent énoncent des solutions non-libres. Ci-dessous nous pouvons observer le commentaire d'un utilisateur, rédigé pour convaincre une personne qui voulait arréter d'utiliser gNS :

> *Installe Ubuntu,[...] ensuite tu pourras suivre l'évolution du logiciel libre et migrer petit à petit vers des alternatives libres, un bit après l'autre...ok, Ubuntu n'est pas libre, mais c'est plus libre que Windows.*
> Je comprends ce que tu essaies de faire, cependant le message que tu fais vraiment passer est : "pas de problème si tu utilises des logiciels non-libres pour le moment ; quand le logiciel libre n'aura plus de différences fonctionnelles avec le non-libre, alors reviens". Cela rend artificielle l'éthique du logiciel libre et ne nous mène nulle part.[5]

Dans cet exemple, le développeur reproche à l'utilisateur de proposer un logiciel non-libre (une distribution plus intuitive de Linux) comme un moyen pour arriver à une utilisation exclusive du Libre. Ceci est à contresens de l'effort civique radical de la communauté. Dans ce sens, la communauté gNS se caractérise par une forte posture politique et par des critères traditionnels, c'est à dire, un idéal réalisé par une action commune, contre un ennemi commun déclaré. De fait, les quatre libertés et la philosophie orginelle du projet GNU sont les bases de ce mouvement technologique de création

[5]" then you might follow the evolution of Fsoft and gns, change to free alternative, one byte after the other...ok, ubuntu is not free, but it's freer than Windows." / "I see what you're trying to do, but the message you actually send out is : "it's ok to use non-free software for now ; once free software has caught up and there is no functional difference between free and non-free anymore, then you can switch back". This hollows out the ethics of software freedom and doesn't get us anywhere"

d'un système opérationnel et d'une plate-forme de développement de logiciel libre. Ceci voulant se maintenir comme une alternative contre toutes les formes non-libres de code. Cependant, à cette fin, il y a un "sacrifice" technique à ne pas pouvoir utiliser certains outils basiques du monde virtuel contemporain. Par exemple, assister à des vidéos en ligne ou accéder à certains sites internet "dynamiques", ou encore accéder à plusieurs formats basiques d'applications de bureau. Dans ce sens, les participants à ce projet prétendent activer un civisme, à travers des pragmatiques de la liberté radicales en essayant d'améliorer le coût technique du développement d'une alternative entière dont quiconque peut profiter.

3.1.3 Logiques d'inversion

Le système de propriété intellectuelle qui structure le monde du logiciel et ses conflits est basé sur des réglements, des lois, des conventions nationales et internationales (par exemple le DMCA – *Digital Millenium Copyright Act*), que des licences particulières invoquent. Une technologie est fermée ou restreinte quand elle se réfère à des concepts de copyright ou de droits d'auteurs que l'utilisateur accepte en installant le programme. L'objectif du projet GNU et de la licence GPL a été d'utiliser cette même légitimité juridique pour défendre le contraire, c'est à dire, la non-appropriabilité du code mis à disposition. C'est une logique d'inversion des défenses proprétaires pour construire les défenses du Libre.

De cette manière, les préoccupations légales sont au centre de l'activité de la commuauté gNewSense et pour cela il existe des "contrôleurs des 4 libertés" (*4Freedom Checkers*) en plus des catégories traditionnelles de développeurs (codeur, aide à l'utilisateur, gestion des bugs, etc). Ces développeurs surveillent le système et tous les programmes disponibles pour vérifier l'absence de failles légales permettant à un code restreint d'être installé sur la machine. Cependant, l'exigence d'une licence compatible avec le GPL n'est pas suffisante pour cet objectif, et les développeurs cherchent aussi à déterminer si les programmes ne sont pas liés à des sources secondaires restreintes. Encore une fois, les implications d'un tel engagement apparaissent particulièrement au contact des utilisateurs. En témoigne la réponse d'un développeur à un utilisateur souhaitant installer un émulateur Windows (Wine, sous licence GPL) pour pouvoir utiliser ses jeux (propriétaires) :

> Un des points clés pour comprendre gNewSense et l'ensemble du logiciel libre est de comprendre que leur objectif légal est d'être complétement libre. Alors que j'ai besoin de faire des recherches pour savoir si Wine est libre ou non, le logiciel libre n'est pas fait

> pour donner un moyen gratuit pour excécuter toutes tes applications Windows. C'est un mouvement politique, social et technologique pour un logiciel libre.[6]

En soulignant ce qui constitue la base politique et sociale qu'évoque le développeur, nous mettons en lumière la recherche pour une homogénéité juridique inédite et absolue. Ainsi une bonne partie des débats au sein des listes de mails de développement de la distribution est concentrée sur les aspects légaux des logiciels, alors que les aspects techniques, quand ils ne traitent pas d'une technologie alternative, sont laissés aux bons soins de la distribution-mère du système (Ubuntu). Séparer ce qui est libre de ce qui ne l'est pas est la principale logique d'organisation du système gNewSense. Les logiques de liberté amènent les programmeurs à choisir quels logiciels peuvent être installés et lesquels ne peuvent pas l'être. Ainsi, nous illustrons une inversion du radicalisme propriétaire pour un radicalisme du Libre, et, plus encore, une ignorance délibérée de tous les "compromis" qui se rencontre dans le milieu technologique de l'Open-source.

3.1.4 Logiques de collaboration

Un espace de collaboration sans restrictions est l'objectif final de la GPL ; et les possibilités de collaboration instaurées par la communauté gNS sont en même temps absolues et fermées. De fait, d'un côté, le potentiel de dynamique participative au sein du système est infini car tout le code utilisé est libre, accessible, ouvert, modifiable dans les termes de la GPL et de ses licences semblables. D'un autre côté, l'espace ainsi créé ne profite pas à d'autres projets dont les conditions d'appropriabilité sont distinctes. Pour cela, en considérant l'extrême variété des natures juridiques des codes dans le développement des technologies, le domaine de la GPL est totalement hermétique à un grand nombre d'innovations. Celles-ci doivent toujours être recodifiées (quand elles ne sont pas brévetées) au sein d'un projet qui corresponde aux critères de la communauté.

De plus, l'idéologie du projet à propos de ses possibilités de collaboration appartient à deux délais temporels. Ainsi, nous comprenons qu'à court terme, il y a sacrifice au nom d'une utopie, une idée de liberté absolument non restreinte qui empêche certaines possibilités de collaboration et, ainsi, permet

[6]"one of the key points about understanding gNewSense and all free software, is to understand that its legal goal is to be completely free software. While I need to do more reasearch into weather or not Wine itself is, Free Software isn't about giving a free (monetary wise) way to runs as many windows programs as you can. It's a political, social, and technological movement for Free (as is Freedom) Software"

des échanges de codes seulement au sein du domaine de la GPL. Ce compromis est entièrement dédié à l'idéal du projet GNU, afin qu'à long terme soit réalisée sa conception de collaboraiton, absolue et non-restreinte. Ceci représente un positionnement réellement activiste au sein des communautés du Libre. Moyens et fins se mélangent pour donner l'objectif de ce mouvement, reflétant ainsi l'idéologie proposée. De cette manière, la contradiction qui souligne le manque d'ouverture comme une restriction à la collaboration n'atteint pas ses actants concentrés sur un objectif majeur.

3.2 Les pragmatiques d'ouverture : la communauté Samba

Une des principales difficultés affrontées par les systèmes Linux au sein des institutions, a été son manque de compatibilité avec les protocoles de codes fermés. De fait, avec la multiplication des stations Windows, toutes les institutions sont passées à des protocoles au sein de réseaux internes qui permettent aux ordinateurs de communiquer. Un tel environnement a été difficile pour les systèmes Linux, serveurs comme clients, pour gérer de tels protocoles, ou même, simplement communiquer avec eux. C'est dans l'intention d'atténuer cette carence spécifique qu'à été créé le projet Samba, un projet maintenant ancien et réussi au sein du Libre.

Initié comme un projet de doctorat par un étudiant australien, Andrew Tridgell, en 1991, le logiciel Samba a permis peu à peu aux systèmes basés sur Unix (Linux, BSD, et autres) de dialoguer sur les réseaux avec les systèmes et serveurs Windows. En pratique, cela a permis, entre autres choses, aux clients Windows d'interagir avec les serveurs Unix, notamment pour les services d'impression et de partage de fichiers. Licencié sous GPL, le projet est devenu un standard dans toutes les branches du Libre pour gérer les protocoles de réseaux hybrides.

Les analyses proposées ici sont réalisées à partir de matériaux extraits principalement du Forum[7], des listes de mails et du principal canal IRC[8] de la communauté.

3.2.1 Logiques de transgression

Les logiques de transgression sont présentes au sein de la communauté Samba du simple fait que sont elles qui ont fondé ce projet. Afin de réussir à

[7] `http://www.nabble.com/Samba-f13150.html`

[8] #samba@irc.freenode.net :6667

communiquer avec les protocoles de Microsoft, qui sont fermés, il a été nécessaire d'utiliser les méthodes de la rétro-ingénierie. Étant fermés, les protocoles se présentent aux développeurs comme une "boîte noire" dont les caractéristiques peuvent être découvertes uniquement par l'analyse de ses entrées et sorties. En utilisant des *packet sniffers*, les développeurs utilisent des outils communs à l'exploitation de réseaux pour "sentir" le comportement des protocoles fermés et ainsi déterminer quel code permettra de communiquer avec eux. Samba explore en fait un des cas où cette pratique est permise, c'est à dire, quand elle est réalisée afin de permettre une interopérabilité entre des normes divergentes.

Une telle activité s'illustre par l'exigence d'une compétence technique avancée, et représente un défit attrayant pour ses actants : déconstruire un secret propriétaire et le faire fonctionner avec une technologie libre. Les références communes au sein du *hacking underground* sont présentes dans les discours des développeurs dans la mesure où elles ne dépassent pas les limites de ce que la loi permet, car, étant un projet ouvert et très utilisé, les interactions des utilisateur-développeurs se déroulent au vu de tous.

Cependant, dans un contexte plus général d'observation, nous pouvons souligner ici une des caractéristiques des politiques informelles de transgression, c'est à dire, le fait que l'acte transgressif soit très dépendant de l'objet qu'il tente de déconstruire. Si on considère que l'acte de permettre à des protocoles libres de dialoguer avec d'autres propriétaires comme un acte en faveur du Libre, alors l'acte de transgression réalisé reste nécessaire tant que ces protocoles sont maintenus. Dans ce sens, si les motivations des utilisateurs-développeurs qui participent de la communauté Samba sont, en partie, inspirées par les logiques de transgression que son développement nécessite, elles sont alors dépendantes de l'existence de ces protocoles propriétaires. Pour cela, la logique transgressive permet autant aux protocoles propriétaires qu'aux libres de survivre dans un écosystème qui n'est pas entendu de manière revendicative. La pragmatique d'ouverture, dans sa logique transgressive, ne souligne pas la présence de code propriétaire sur le réseau, mais cherche un "droit" à pouvoir communiquer avec eux. L'environnement majeur où interagissent ces codes n'est pas présenté comme étant libre, mais plutôt comme étant naturellement hétérogène.

3.2.2 Logiques de civisme

Les méthodes de rétro-ingénierie se trouvent dans divers projets libres. Il s'agit de "courir derrière" des formats de fichiers ou des protocoles indispensables au fonctionnement basique d'un système. Tout comme Samba cherche à intégrer les protocoles de Microsoft au monde Unix, Open Office "cours

derrière" le format ".doc" et Wine derrière les Interfaces de Programmation d'Applications (API) du système Windows. Le résultat ainsi proposé à l'utilisateur final n'est plus une alternative, comme c'est le cas avec gNewSense, mais une interopérabilité, un code qui permet à des technologies hybrides de communiquer. Étant donné que le but le plus fondamental des technologies de l'information est la communication, un tel objectif d'interopérabilité a beacoup plus d'impact que celui de création d'un alternative autonome. Il s'agit de permettre les compromis.

Cependant, il y a une proposition politique forte derrière l'usage de la rétro-ingénierie, car elle est toujours défendue par ses utilisateurs comme un modèle de développement très fondamental à l'intelligence humaine. Il s'agit de regarder un objet inconnu et de le comprendre à partir de la relation entre ce qu'il produit et ce qu'il a traité. Cette opération est considdérée comme un "droit fondamental" du programmeur, comme de l'être vivant en général. Ne pas avoir l'autorisation d'utiliser cette forme d'intelligence est perçu comme une restriction à une condition très naturelle. Dans ce sens, un développeur web commente :

> Bon, je suis programmeur web, je travaille beaucoup sur des technologies Flash, qui n'est pas libre [...] ce que j'aime dans la philosophie du libre est qu'elle semble défendre mon mode d'apprentissage contre un autre modèle qui semble l'interdire [...] je n'ai jamais été à l'Université, j'ai tout appris par moi même, et pour cela, je manque de concepts généraux... Ce que je sais c'est voir un truc et comprendre comment ça marche, peu importe que ce soit libre ou non... la boîte noire c'est mon quotidien... bon... il y a le "bidule" qui fait des trucs et en reçoit d'autres, comment marche-t-il ? Comment je fais pour qu'il traite une chose ou en produise une autre ? c'est comme un bébé qui apprend à parler ou un ado qui apprend à baiser : ça marche ? oui, je garde. ça ne marche pas ? je dégage... ça marche plus ou moins ? j'ai besoin de jeter un coup d'oeil, etc... et comme ça j'avance, j'assemble des solutions, des trucs pour assembler de nouveaux savoirs, des outils, etc ...

Si l'interdiction d'une telle pratique était un jour banalisée au monde de l'informatique, il y a l'idée qu'une cognition très fondamentale aux programmeurs ne pourrait plus s'exprimer, qu'une rationalité n'aurait plus le droit de se déclencher. Pour l'instant le DMCA considère cette pratique comme acceptable à des fins d'observation ou d'éducation. Cependant les acteurs propriétaires soulignent le fait que beaucoup d'opérations de piratage sont réalisées par ces techniques, et, pour cela, souhaitent les voir interdites ou

bien fortement réglementées.

Ainsi, les communautés libres comme Samba, qui pratiquent ce modèle technique, défendent profondément ce droit en soulignant sa valeure sociale et politique comme défense d'une modalité primaire de l'entendement humain.

3.2.3 Logiques d'inversion

Pour ne pas être une alternative mais un effort dans le sens d'incorporer les réalisations du monde propriétaire, la communauté Samba n'opère pas une logique d'inversion au-delà de ce que sa licence (GPL) réalise déjà. Pour cela, ce sont les autres logiques examinées ici qui doivent être considérées avec le plus grand soin pour comprendre la pragmatique d'ouverture.

3.2.4 Logiques de collaboration

Étant une communauté très ouverte aux contributions et ayant des utilisateurs-développeurs des mondes libres et propriétaires en son sein, la communauté Samba incarne bien les logiques de collaboration qui se rencontrent dans le monde open-source. L'interopérabilité des technologies et les logiques de collaboration apparaissent comme l'ultime objectif de la communauté Samba. De plus, bien que l'entrée permise aux technologies restreintes afin qu'elles puissent communiquer avec les Libres soit une réalisation qui puisse être critiquée du point de vue de la philosophie GNU-GPL la plus radicale, elle est très encouragée par l'aile pragmatique du mouvement du Libre.

On trouve derrière une telle idée une interprétation différente des limites entre ce qui peut rester fermé et ce qui doit être ouvert. La différence apparaît à un niveau technique où, alors que GNU cherche une solution homogène libre, une projet comme Samba cherche à créer un "possible" du Libre, c'est à dire l'opportunité qu'un code libre puisse communiquer avec n'importe quel autre. Le libre est alors entendu comme un ensemble homogène certes, mais capable de dialoguer avec d'autres ensembles de natures différentes, et pour cela, voir sa pragmatique d'ouverture renforçée. Un exemple évoqué par un des fondateurs de Samba explore cette perpective à propos d'une des préoccupation actuelle du mouvement du logiciel libre, qui est d'offrir une alternative aux protocoles propriétaires de VoIP (Skype, Google Talk, entre autres) : "J'espère seulement, qu'éventuellement, les développeurs de logiciels libres réussissent à s'accrocher au réseau Skype et à dialoguer avec ses protocoles. Après tout, il y a d'excellents précédents de cela avec d'autres logiciels libres..."[9] [Allsion, 2005].

[9]"I just hope that eventually Free Software developers can work out how to hook into

Comme nous pouvons l'observer la référence souligne la faiblesse du projet Ekiga, alternative peu utilisée chez Skype, le logiciel phare du domaine de VoIP. Alors que Ekiga offre un logiciel libre pour utiliser ses propres protocoles libres, Jeremy Allison propose comme solution l'utilisation de la rétro-ingénierie pour permettre à un logiciel libre d'interagir avec les protocoles propriétaires de Skype. Nous voyons ainsi que les logiques de collaboration développées par les pragmatiques d'ouverture promeuvent non pas un compromis du Libre envers le propriétaire, mais une réinterprétation du domaine du libre par rapport aux protocoles de communication.

3.3 Les pragmatiques de sécurité : les communautés BSD

Les pragmatiques de sécurité illustrent l'importance toute particulière donnée aux performances d'une technologie informatique. De fait, parmi les préoccupations d'un programmeur, comme la rapidité, l'interopérabilité ou la légèreté d'une technologie, il existe aussi la notion de sécurité. Un logiciel, comme tout "système", est cassable et possède des failles qui sont autant d'opportunités pour une tierce personne d'exploiter et de modifier ainsi son comportement sans se soumettre aux règles de l'agorithme principal. Avec la diffusion à grande échelle de l'Internet à partir des années 1990, on a souligné l'importance des problématiques de sécurité afin de limiter la diffusion de virus et l'exploitation des réseaux privés.

De fait, les technologies libres ont toujours prétendu être plus sûres car ouvertes, c'est à dire, que chaque faille restant plus apparente, elle est plus rapidemment communiquée et réparée. Indépendemment des ambitions du Libre, et plus par défi technologique, les systèmes Unix basé sur les noyaux BSD, particulièrement destniés aux serveurs, ont toujours développé la sécurité et la stabilité.

Les communautés BSD sont celles qui se sont formées aux alentours de la distribution d'Unix développée par l'Université de Berkeley (BSD – Berkeley Software Distribution), entre 1977 et 1995. Après cette date, elles se sont divisées en trois familles principales : FreeBSD, OpenBSD et netBSD. Nous nous concentrerons principalement sur la communauté FreeBSD, qui est plus large (77% des systèmes BSD en usage en 2005[10] et celle de OpenBSD qui ont des projets connexes très productifs (OpenSSH, OpenSSL).

---

the Skype network and inter-operate with the Skype protocols, after all, there are good precedents for that with other Free Software…"

[10]Fonte : `http://www.bsdcertification.org/downloads/pr_20051031_usage_survey_en_en.pdf`

Encore une fois, les analyses proposées ici sont basées sur des matériaux principalement extraits du Forum[11] , des listes de mails et des canaux IRC[12] de la communauté.

3.3.1 Logiques de transgression

La stabilité que les technologies BSD mettent à disposition nécessite de la part des utilisateurs-développeurs beaucoup de participation dans le domaine de la sécurité. Pour cela, beaucoup de développeurs sont connectés aux mêmes circuits d'information qui diffusent les failles de systèmes. La logique profonde de la sécurité en informatique répond à un échange entre attaquants et défenseurs.

D'un point de vue représentatif, les attaquants veulent trouver des failles pour les explorer, mais souhaitent un système sûr afin de l'utiliser. Les défenseurs veulent aussi trouver les failles afin de les réparer et pour cela souhaitent dialoguer avec les attaquants. En résumé, chacune des parties cherche ce que l'autre trouve, et pour cela, les hautes exigences de sécurité des systèmes BSD rapporche ses développeurs des champs virutels du *hacking underground*. Cependant, l'objectif qui motive les développeurs BSD dans le domaine de la sécurité est la stabilité du système, pour cela, la transgression des technologies est uniquement un moyen indirect d'y parvenir.

L'inversion opérée au niveau des logiques de transgression est la base du processus qui permet de rendre sûres les technologies BSD. L'acte transgressif de recherche de failles est reproduit intégralement par les développeurs, mais son objectif est inversé du fait d'être destiné à la correction de failles. Un exemple est la procédure d'"audit" (*OpenBSD code auditing*) réalisée par la communauté OpenBSD sur tous les logiciels mis à disposition avec la distribution. Un telle procédure consiste à lire, ligne par ligne, l'ensemble des codes, en cherchant les failles potentielles de ceux-ci, ce qui est exactement ce que ferait un *cracker* mal intentionné afin de pouvoir les exploiter.. Le *code auditing* va donc au-delà du simple échange d'information, mais permet une exploration sûre car libérée de la collaboration avec les *crackers*. Dans ce sens, la communauté BSD met en avant l'argument qu'avec un tel mode de développement elle a réussi à mettre à disposition un système qui n'a pas souffert de failles de sécurité pendant 5 ans[13]. Une telle préoccupa-

[11] `http://forums.freebsd.org/`

[12] #freebsd@irc.freenode.net :6667

[13] Jusqu'en juin 2002 le slogan de OpenBSD était : "Cinq années sans failles dans l'installation par défaut !". En 2007 le slogan était : "Seulement 2 failles dans l'installation par défaut depuis un sacré bout de temps !". Pour avoir un point de comparaison, il y a des actualisations de sécuritié chaque mois pour des sytèmes comme Linux ou Windows.

tion détermine en grande partie les logiques de civisme (contributions) et de collaboration des pragmatiques de sécurité des communautés BSD.

3.3.2 Logiques de civisme

La diversification des activités sur l'Internet, particulièrement celles nécessitant l'échange d'argent, comme c'est le cas des achats en ligne, ou celles de consultation de données sensibles (compte bancaire, par exemple) fut permis petit à petit par l'utilisation de techniques de cryptographie à grande échelle. Une des plus grandes contributions dans ce domaine a été mise à disposition par la communauté OpenBSD, dont les projets OpenSSL et OpenSSH ont "libéré" les premiers efforts dans ce sens. De fait, le projet OpenSSH est adopté dans toutes les distributions Unix, c'est à dire sur à peu près tous les systèmes qui ne sont pas Windows et permet à deux systèmes de dialoguer par une connexion protégée et cryptée. Cependant, le projet le plus répandu est OpenSSL, qui prétend offrir la même sécurité pour les connexions Web. Sans doute, beaucoup de connexions qui se font sur le web nécessitent un type de protection, comme c'est le cas pour des achats ou pour la divulgation de mots de passe, afin que les données ainsi transférées n'apparaissent pas "en clair" à la "lecture" du réseau.

Il y a une dimension politique et sociale à de tels actes, proche de celle des premiers pas de la cryptographie. La sécurité et la privacité sont considérés par les actants de ces communautés comme une nécessité sociale, une carence. Le code ainsi produit est un acte politique et social dans le sens de recouvrir ces nécessités. Un tel effort est profitable à toute la société connectée. Une communication sécurisée protège le commerce et des transactions financières, de la dite "piraterie" de contenu intellectuel et les réseaux gouvernementaux, etc... Il y à la défense d'un droit à la privacité qui définit ses termes sans limites d'objets à protéger. Alors que ces fonctions dans le monde politique traditionnel peuvent apparaître comme relevant du domaine régalien qui a obtenu le monopole des mesures de sécurité, au moins en terme de légitimité, et définissent les limites du droit à la privacité. Dans le cas de la criptographie, et des outils libres comme ceux des communautés BSD, les possibilités de sécurité et de privacité sont mises à disposition d'acteurs indépendants qui gagnent leur légitimité du fait d'être de commmunautés "libres". Ainsi, le pouvoir politique traditionnel peut uniquement interagir avec elles par le moyen de restrictions autoritaires et non de mises à disposition ou de monopoles des outils.

3.3.3 Logiques d'inversion

La licence BSD peut favoriser une collaboration sans restrictions permettant l'appropriation de son code par des projets propriétaires et fermés ; pour cela les commuautés BSD ne peuvent pas être identifiées aux logiques d'inversion comme le peuvent celles proches des domaines GNU-GPL.

3.3.4 Logiques de collaboration

Les communautés BSD ne sont pas explicitement ouvertes comme celles de Samba ou de gNewSense. Quand on entre sur le site internet du projet FreeBSD il n'y a pas un lien ou une annonce explicite invitant les utilisateurs à contribuer ou à participer. Les impératifs techniques propres à ces communautés exigent un haut degré de compétence souvent pointé du doigt comme "élitiste" par les autres communautés libres. Les actants considèrent que l'exigence technique qui fonde le projet leur interdit d'accepter les amateurs dont le travail peut être perçu comme mal fini et probablement de piètre qualité eu égard à leurs exigences de stabilité.

Ce "professionnalisme" attire tout particulièrement les investisseurs privés car il propose des conditions d'appropriation très avantageuses pour les entreprises qui développent des produits fermés. La licence BSD permet l'utilisation d'un code sous son empire sans aucun retour pour les communautés. Beaucoup d'exemples de collaborations ainsi déterminées ont donné des produits reconnus dans le marché des nouvelles technologies, parmi lesquels : les consoles de jeux Xbox (FreeBSD – Microsoft) et PSP (NetBSD – Sony), MacOSX (FreeBSD – Apple), entre autres.

Les codes BSD sont aussi réutilisés pour des projets libres, et incoporés au système Linux par exemple. Cela positionne les communautés BSD comme plate-formes possibles de collaborations entre les logiques propriétaires et libres, et leurs utilisateurs-développeurs aiment souligner qu'une telle caractéristique favorise avant tout l'excellence technique du système et permet à un environnement diversifié de se maintenir. Dans ce sens, un développeur du Forum FreeBSD illustre : "C'est le moment de comprendre que FOSS est avant tout versatil. Tu ne voudrais pas que Linux soit BSD, que BSD soit MacOSX, etc... Comme ça on a le CHOIX. Quelque chose qui est "bon à tout" ne peut pas, par définition, être bon."[14]. De cette manière, nous illustrons une logique différente dans l'interprétation de ce qu'est le logiciel libre : la versatilité d'un environement dont la diversité technique et légale permet

[14]"It's time to understand FOSS is versatile. You wouldn't want Linux to be BSD, BSD to be OS X ... etc. This gives us CHOICE. Something that is "good for everything" cannot be good by definition.".

à des solutions différentes de s'adapter à des nécessités hétérogènes.

3.4 Comparaison et analyse des caractéristiques des pragmatiques

FIG. 3.1 – Tableau récapitulatif

	Pragmatique de **liberté** (*gNewSense*)	Pragmatique d'**ouverture** (*Samba*)	pragmatique de **sécurité** (*BSD*)
Logiques de **transgression**	0	Retro-ingenierie	Recherche de failles
Logiques de **civisme**	Alternative entière et autonome aux outils non-libres	Interopérabilité des technologies	Sécurité et stabilité du dispositif
Logiques d'**inversion**	Espace juridique propre, protégé par la même légitimité légale que le domaine propriétaire	0	0
Logiques de **collaboration**	Communauté ouverte ; Collaboration absolue en espace hermétique	Communauté ouverte ; Collaboration non-limitée par le domaine du Libre	Versatilité ; Communauté relativement fermée ; Potentielle exploitation commerciale unilatérale

En comparant les pragmatiques de la programmation exposées dans la figure 3.1 (p.66) nous pouvons observer que ces trois communautés interprètent différemment leur participation au mouvement du logiciel libre et que ces différences s'expriment en premier lieu dans les choix réalisés par rapport au logiciel objet de la communauté.

Cette diversité technique souligne pour ses actants des représentations différentes de ce que signifie être libre et ouvert, ce qui les motive à agir dans ces projets et, enfin, avec quel environnement technologique majeur le Libre interagit-il. De telles caractéristiques constituent la personnalité sociale et politique de chaque communauté et les différencient les unes des autres.

Les choix réalisés forment le squelette politique d'une communauté, de son code et de son action dans le champ technologique. Que ce soit autour d'une préoccupation esthétique ou d'une responsabilité par rapport à un potentiel régulateur, les mêmes génériques décisionnaires sont mis en place dans le processus communautaire de création et de promotion du logiciel.

Au-delà, quelques éléments apparaissent à la lecture du tableau récapitulatif (Figure 3.1). Premièrement, les logiques de transgression et d'inversion semblent s'exclure l'une et l'autre. Dans le cas des pragmatiques de liberté, les logiques de transgression sont peu recommandées au sein du discours de la communauté GNU-GPL du fait qu'elles peuvent-être une porte d'entrée à certains compromis par rapport à son idéologie. Pour leur part, les pragmatiques d'ouverture et de sécurité développent notablement les caractéristiques transgressives de leurs modèles de développement afin de permettre l'entrée de l'objet transgressé dans le code du logiciel. Il s'agit, dans le cas de Samba, des protocoles de communication propriétaires ; et dans le cas de BSD, la défense contre les failles de sécurité.

En partie, ces caractéristiques des logiques de transgression déterminent celles du civisme. Alors que l'absence de transgression dans la pragmatique de liberté produit la nécessité de construire une alternative entendue comme absolue, la présence de telles logiques dans les pragmatiques d'ouverture et de sécurité soulignent l'effort dans le sens d'une performance technique plus pragmatique. Il s'agit, dans le cas de Samba, d'une contribution à l'interopérabilité des systèmes Unix avec le monde propriétaire ; et dans le cas de BSD, de la mise à disposition d'outils sécurisés et stables pour servir l'activité de tout un chacun sur le réseau.

Nous pouvons donc observer que les logiques d'inversion caractérisent très spécifiquement le domaine GNU-FSF. C'est à dire que les logiques d'inversion propres aux pragmatiques de liberté comme on les observe dans la communauté gNewSense sont le monopole d'un software qui se veut exclusif. En ayant cherché ses défenses contre le monde propriétaire dans les mêmes outils légaux, c'est à dire les licences, le domaine GNU que le projet gNS vient systématiser, cherche à inverser les propositions des systèmes opérationnels propriétaires comme hybrides.

Enfin, les logiques de collaboration semblent offrir un bon reflet de la sphère politique de chacune des communautés, comme un résultat concret de sa capacité à interagir avec son environnement technologique. Pour cela, alors que les pragmatiques de liberté promeuvent une logique de collaboration du domaine de l'utopie, c'est à dire, absolue, mais dans un espace restreint à ne faire aucun compromis, les pragmatiques d'ouverture et de sécurité, elles, interagissent de manière conséquente avec leurs environnements respectifs. Pourtant ces interactions se différencient de manière significative quant à

leur nature. La pragmatique d'ouverture fait de la collaboration un objet du software qu'elle développe, l'objet de la collaboration est attiré dans l'effort technologique du Libre pour que celui-ci s'en voit accru. Différemment, la pragmatique de sécurité met en avant une collaboration très forte avec n'importe quel type d'acteur tant que le "professionnalisme" du système est maintenu.

Nous pouvons observer alors qu'au sein d'un même mouvement du logiciel libre, interagissent des oppositions radicales de visions politiques et sociales, en termes de moyens de développement, d'objectifs établis, de contributions réalisées et de collaborations constituées. Cet ensemble hétérogène véhicule cependant une même figure lorsqu'il se relationne avec sa sphère politique majeure, comme mouvement uni, ou du moins désigné comme tel. Ces branches réformistes et progressistes, ou révolutionnaires sont mélangées dans une même entité sociale recueillie par le sens commun sous le nom générique de Mouvement du logiciel libre et open-source (FOSS). Pour cela, une des hypothèses qui peut être émise ici est que l'objet traditionellement entendu comme "mouvement" pourrait en fait être perçu comme un public. Dans ce sens, il est intéressant de mentionner l'effort, peut être contradictoire, des communautés du Libre pour exprimer un discours sur des sujets concurrents ou opposés, par exemple Libre et/ou Ouvert, pour que soit maintenue une même identité commune, celle d'un public diversifié, dont les interactions structurent un panorama technologique des logiciels. C'est ainsi que témoigne le fondateur du projet gNewSense, Paul O'Malley, lors d'un entretien :

> Les développeurs de logiciels de divers domaines, c'est à dire autant de Linux que de BSD, utilisent des licences qui respectent les quatre libertés. Il y a beaucoup d'autres domaines, mais ces deux là suffisent aux objectifs de cette analogie. Chacun de ces champs permet facilement à chacun de participer, et au delà, de créer de l'interpolation, au sens où l'entendent les sciences sociales, justifiant auprès des personnes qui les rejoignent qu'il sont dans le bon camp, grâce à des philosophies internes consistantes. Celles-ci sont très similaires, et ne se différencie que dans certains cas marginaux. Chacune proclame être la plus valide, néanmoins leurs différences font qu'elles sont fortement distinctes.
>
> Elles sont comme des jumeaux qui partagent beaucoup du même ADN, néanmoins, bien que séparées, ce sont des entités légitimes du logiciel libre. Un preuve positive de ceci est que si vous suivez chacune d'entre elles, une communauté a son travail mis dans MacOSX et l'autre dans le projet Debian, et qui peut éventuellement

s'orienter vers gNS[15].

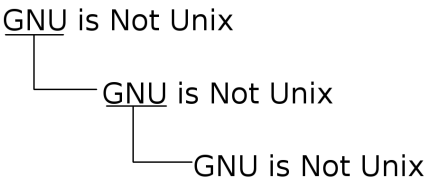
Ainsi, se présente à la compréhension de l'observateur, non pas un mouvement politique homogène, mais un public diversifié dont la dynamique se constitue à partir d'un ensemble de mouvements concurrents, et parfois, opposés. Les philosophies développées par chacun des camps du Libre sont construites en fonction des projections et représentations que les différentes communautés produisent à partir des choix qu'elles réalisent en codifiant leur logiciel. Cependant, un ensemble commun (l'"ADN") contribue à promouvoir un modèle de développement plus abstrait et plus partagé.

De toute manière, qu'elle soit hybride ou absolue, une meme notion de liberté à laquelle doivent répondre les licences ("les quatres libertés") est entendue comme acquise. Peu importe qu'une communauté soit "élitiste" ou bien "éducative" face aux utilisateurs-développeurs, il y a une même idée de leur rôle, comme quelque chose de contributif à un effort de développement libre et ouvert. Enfin, pour autant que les applications faites de ces logiciels puissent être radicalement différentes, restant libres ou devenant propriétaires, l'ensemble primaire de logiciels reste accessible dans les mêmes termes par tout un chacun. Ces points communs constituent un public, dont les modalités communautaires produisent des mouvements qui font la dynamique politique du logiciel Libre.

[15]"Software developers in the various camps, that is the developers of GNU/Linux systems, and developers in the various BSD camps use licences which respect the four freedoms. There are many other camps but these two will suffice for the purposes of this analogy. Both camps make it very easy for one to join and upon joining make it easy for one to interpolate in a social science way, justifying to the person joining their particular camp, and that most certainly they are in the right place by having internally consistent philosophies. They are very similar, and the edge cases they differ. Both claim to have the ultimate validity, however they are by virtue of their differences very distinct entities. / They are like twins sharing a lot of DNA, in so far as they are legitimate free software entities albeit separate. Proof positive is when you get down streams of both, one community had their work put into OS X and the other into Debian project, the latter of which eventually turns into gNewSense."

Conclusion

On m'a raconté une plaisanterie que Richard Stallman à l'habitude de faire durant ses nombreuses apparitions publiques en congrès ou séminaires, afin de promouvoir le logiciel libre. Il va jusqu'au tableau ou prend une feuille de papier et commence à dessiner :



"Le mouvement du logiciel libre est tellement malin que même son nom est récursif !" s'exclame-t-il au milieu des rires des ingénieurs, sympathisants et amateurs du Libre.

Il y a une pensée contemporaine qui cherche à concevoir la technique comme un simple outil. Ainsi, le comptable, le mécanicien, de manière générale le *technicien* possèdent une pratique hermétique qui en sert une autre, celle de la *pensée* politique, philosophique, sociale, économique. On retrouve les origines de telles idées dans la philosophie grecque antique et sa conception classique du savoir comme un *dedans* qui doit rester indépendant de son *dehors*, la technê. La technique, elle, ne participe pas de la recherche du bonheur, de la beauté, du juste ou du vrai, elle la sert.

Surgit alors l'image platonicienne du sophiste, qui utilise la pensée comme une technique pour rendre n'importe quelle question indiscutable, souvent à des fins jugées immorales. De nombreuses historiographies de cette époque, cependant, ont montré que le courant sophiste avait été discrédité des piliers de l'histoire des idées et qu'il s'agissait en fait d'une philosophie à part entière, pragmatique et rationnelle, qui privilégie l'analyse des situations, des lieux, des évènements et des langages de manière concrète et non comme une fin en soi.

Si ce n'est pas une décision politique qui conduit le mouvement technico-scientifique actuel et que nous refusons l'apriori d'immoralité ou d'inutilité d'une technê autonome, qu'est-ce qui conduit le mouvement des réseaux et des matériaux, qui construit notre monde déjà dépendant de lui ? On trouve des éléments de réponses dans l'oeuvre de Christopher Kelty et son anthropologie du Geek à travers l'étude des significations culturelles du mouvement du Libre. Selon lui, la participation collaborative et ouverte de nombreux individus a permis à un public *récursif* légitime de se constituer. Tout comme la fonction récursive factorielle, fonction mathématique (figure 3.2) qui multiplie un nombre par les entiers qui le précèdent et ainsi produit un nouveau nombre plus grand, le public identifié par Kelty réalise son présent en invoquant ses éléments déjà existants.

FIG. 3.2 – La fonction factorielle

$$n! = \prod_{i=1}^{n} i = 1 \times 2 \times 3 \times \cdots \times (n-1) \times n \tag{3.2}$$

Le logiciel est le concept que les communautés libres viennent invoquer récursivement. Le logiciel invoque le logiciel, qui invoque le logiciel, qui invoque le logiciel. Et ainsi produit des programmes, des formats, des protocoles qui participent au mouvement technologique contemporain.

Nous pourrions décrire un tel mouvement comme l'expérience vécue de la pensée pragmatique, selon laquelle, "l'effort n'est pas de pratiquer l'intelligence, mais d'intellectualiser la pratique"[Dewey, 1929]. Il s'agit d'un refus de la recherche de la certitude par le moyen de catégories pré-établies par rapport au contexte d'analyse présent. Á un niveau plus épistémologique, il s'agit d'une position anti-réificative, pour laquelle le concept et la théorie ne sont pas deux objets indépendants, mais des abstractions, dont le produit revient à l'expérience.

Dans ce sens, les premières lignes du *Hacker Manifesto* énoncent : "toutes les classes sont effrayées par cette abstraction implacable du monde, dont les fortunes dépendent encore". Néanmoins, une "classe", la "classe hacker", se positionne différemment : "Nous sommes les hackers de l'abstraction. Nous produisons de nouveaux concepts, de nouvelles perceptions et sensations, hackés à partir de données brutes". Ici, dans une métaphore marxiste d'un monde divisé en classe, la "classe hacker" se différencie par une relation inversée à l'abstraction, comme si elle ne subissait pas sa cognition, mais la produisait à partir d'un traitement conscient des choses en leur état brut.

Comme nous l'avons observé dans le premier chapitre de cette étude, les

alternatives offertes par le logiciel libre se sont constituées par le traitement récursif d'un matière première, Unix d'abord, les projets GNU, BSD, Mozilla, ensuite, afin de constituer un ensemble hétérogène en termes de technologie, de règlement et de sphère politique et sociale.

L'importance donnée à la figure de l'utilisateur-développeur dans le second chapitre nous a permis de trouver les mêmes termes au sein de l'éthique et des significations culturelles du hacking. Dans ce sens, nous trouvons des exemples dans les travaux de Gabriella Coleman, qui nous montre comment "l'agnosticisme politique" des communautés du Libre a permis aux individus de réinterpréter leur liberté technique et éthique dans un contexte digital. L'évolution, l'acte, le mouvement ne s'opèrent pas selon l'idée, le présupposé, des objectifs politiques préétablis, mais par la pratique d'un présent. Ce présent se réalise dans un acte, celui de programmer, dont les pragmatiques appartiennent tout autant à une responsabilité légale, de réglementation de l'espace digital, qu'à une responsabilité artistique, de maintenir et promouvoir une esthétique forte et élégante.

La relation à l'information qui nait de cette pragmatique de la programmation se structure dans les spécificités des communautés que nous avons tenté de souligner chapitre trois. Ainsi, les pragmatiques de l'acte de programmation et les logiques culturelles du programmeur interagissent pour souligner les points caractéristiques des communautés gNewSensense, Samba et BSD, par les priorités qu'elles donnent dans leurs processus de traitement de l'information. Qu'il s'agisse de préoccupations de liberté, d'ouverture ou de sécurité, l'acte reste récursif, du fait de toujours traiter les problématiques communautaires à partir du développement des technologies et informations présentes et passées.

Ainsi se constitu un public dont les modalités donnent leurs contenus politiques à ses mouvements qui se différencient et parfois, s'opposent. Ce qui reste commun à ce public, c'est son traitement récursif des technologies pour produire de nouvelles technologies enrichies et modifiées. C'est ce mouvement technologique qui constitue la base de l'imaginaire politique de ses actants. La technique est entendue comme une infrastructure, un ensemble de moyens socialement identifiés et différenciés pour construire des systèmes opérationnels, des protocoles ou des formats de fichiers qui maintiennent un environnement technologique, alors dépendant de plusieurs éthiques dont les actants se sentent responsables.

Néanmoins, pour autant diversifiée qu'elle soit, l'éthique du Libre reste une alternative à un autre modèle, propriétaire, du fait d'exposer les liens entre la morale et la technique, entre l'infra-structure et la superstructure, le système opérationnel et le système social, comme tant d'objets qui doivent construire un public qui gagne sa légitimité en étant ouvert et libre.

BILBLIOGRAPHIE

AALTONEN, Timo e JOKINEN, Jyke. **Demography of Linux kernel developers**. *Online*, 2006. <http://www.cs.tut.fi/~tta/demography.pdf>

AGAMBEN, Giorgio. **Qu'est-ce qu'un dispositif?**. Paris: Payot & Rivages, 2007.

ALLISON, Jeremy. **A free software confessional**. *Online*: The Low Point, 2005. <http://samba.org/samba/news/articles/low_point/column06.html>

ARMENGAUD, Françoise. **La Pragmatique**. Paris: Presse Universitaire de France, 2007.

BAKTHIN, Mikhail. **Rabelais and his world**. Bloomington: Indiana University Press, 1984 (1941, 1965).

_____. **Toward a philosophy of the act**. Austin: University of Texas Press, 1993.

BAR-HILLEL, Yehosua. **Aspects of Language: Essays and Lectures on Philosophy of Language, Linguistic Philosophy and Methodology of Linguistics**. Jerusalem: Magnes Press, 1970.

BARTHES, Roland. **Le plaisir du texte**. Paris: Seuil, 1973.

BAUDRILLARD, Jean. **Mots de Passe**. Paris: Fayard, 2000.

_____. **Le Système des Objets**. Paris: Gallimard, 1968.

BENKLER, Yochai. Coase's Penguin, or, Linux and The Nature of the Firm. *In*: **The Yale Law Journal**, V.112, 2002.

_____. **Observing networked politics: Theoretical and empirical investigations of power and participation in the networked environment**. *In*: Inauguraçao MediaLab, Sciences Po, Paris, 26/06/2009. <http://www.slideshare.net/medialabSciencesPo/yochai-benkler-inauguration-mdialab-sciences-po?src=embed>

BERDOU, Evangelina. **Managing the Bazaar: Commercialization and peripheral participation in mature, community-led Free/Open source software projects**. Tese de Doutorado, London School of Economics and Political Science, 2007.

BERRY, David. The Contestation of Code: A preliminary investigation into the discourse of the free/libre and open source movements. *In*: **Critical Discourse Studies**, n.1, v.1, 04/2004.

BEST, Kirsty. The Hacker's Challenge: active Access to information, visceral democracy, and discursive practice. *In*: **Social Semiotics**, n.3, v.13, 2003.

_____. Beating Them at their own Game: The cultural politics of the Open Software Movement and the Gift Economy. *In*: **International Journal of Cultural Studies**, n.449, v.6, 2003.

BEY, Hakim. **Caos: Terrorismo Poetico e outros crimes examplares**. Sao Paulo: Conrad, 2004 (1985).

_____. **T.A.Z. The Temporary Autonomous Zone, Ontological Anarchy, Poetic Terrorism**. *Online*, 1991. <http://www.hermetic.com/bey/taz_cont.html>

BLACK, Maurice J. **The Art of Code**. Thèse de doctorat en philosophie, Université de Pennsylvanie, 2002.

BOLAND, Eavan. **Code**. Manchester: Carcanet Press, 2001.

BONE, Jeff. **Prelude to Singularity**. *Online*: silk-list@egroups.com, 27/07/2000. <http://netropolis.in/silklist/msg02844.html>.

BORSOOK, Paulina. **The Cyberselfish: A critical romp through the terribly libertarian culture of High**. New York: PublicAffairs, 2000.

BRETON, Philippe. **Une Histoire de l'Informatique**. Paris: Seuil, 1990.

CARR, Nicholas. **Is Google making us stupid**. *Online*: The Atlantic, 07-08/2008. <http://www.theatlantic.com/doc/200807/google>

CASTELLS, Manuel. **A Sociedade em Rede**. São Paulo: Paz e Terra, v.1, 2002.

CERUZZI, Paul. **A history of modern computing**. Cambridge: MIT Press, 1998.

CHAN, Anita. Retiring the Network Spokesman: The Poly-Vocality of Free Software Networks in Peru. *In*: **Science Studies**, n.2, v.20, 2007.

CHOPRA, Samir e DEXTER, Scott. Free Software and the aesthetics of Code. *In*: CHOPRA e DEXTER, **Decoding Liberation** (Chap.III). New York: Routledge, 2007.

COLEMAN, Gabriella. **The (copylefted) Source Code for the Ethical Production of Information Freedom**. *Online*: Sarai.net, 2003. <http://www.sarai.net/publications/readers/03-shaping-technologies/resolveUid/7ccc93d78eee9a2a6fd01a355944bd13>

_____. How free became open and everything else under the sun. *In*: **A Journal of media and culture**, n.3, v.7, 2004.

_____. The Political Agnosticism of Free and Open Source Software and the Inadvertent Politics of Contrast, *In*: **Anthropology Quarterly**, n.3, v.77, 2004.

_____. Code is Speech: Legal Tinkering, Expertise, and Protest among Free and Open Source Software Developers, *In*: **Cultural Anthropology**, a parecer, 08/2009.

COLEMAN, Gabriella e HILL, Mako. How free became open and everything else under the sun. *In*: **A Journal of Media and Culture**, n.3, v.7, 07/2004.

COLEMAN, Gabriella e HILL, Benjamin, The Social Production of ethics in Debian and Free Software Communities: Anthropological Lessons for vocational ethics. *In*: KOCH, Stefan, **Free/ Open Source Software Development**, (Chap. XIII). IGI Publishing , 2005.

COLEMAN, Gabriella e GOLUB, Alex. Hacker Practice: Moral Genres and the Cultural Articulation of Liberalism. *In*: **Anthropological Theory**, n.3, v.8, 2008.

CORRÊA, Marina Sa. Software Livre e a constituiçao federal de 1988. *In*: **Seminario estudantil de produçao acadêmica**, n.1, v.10, 2006.

CURRAN, Andrew. Managing Creativity: The tensions between commodities and gifts in a digital environment. *In*: **Economy and Society**, n.3, v.36, 2007.

DEWEY, John. **Art as Experience**. 1934.

DIANI, Mario e McADAM, Doug. **Social Movements and Networks: Relational approaches to collective action**. Oxford: Oxford University Press, 2003.

DUJARIER, Marie-Anne. **Le travail du consommateur**. Paris: La Découverte, 2008.

DWYER, Tom. Um Salto no Escuro: Um ensaio interpretativo sobre as mudanças técnicas. *In*: **Revista de Administração de Empresas**, n.4, v.29, 10-12/1989.

EARNSHAW, Nicolaas Charles. **The Samba project: transformation of Self through Open Source Software development**. Monographie, Universidade de Sydney, 2004.

ECO, Umberto. **La Production des Signes**. Paris: Librairie Générale Française, 1992 (1976).

ELIAS, Paulo Cesar e MATTOS, Fernando Augusto. Informaçao e software livre no capitalismo contemporaneo. *In*: **Revista Digital de Biblioteconomia e Ciencia da Informaçao**, n.1, v.5, 2007.

ESCOBAR, Arturo. Other Worlds are (already) possible: Cyber-internationalism and post-capitalist cultures. *In*: **Revista TEXTOS de la cibersociedad**, n.5, 2003.

FABERNOVEL. **Google's key success factors**. *Online*: FaberNovel, 2008. <http://www.fabernovel.com/sites/default/files/Google_KSF_en.pdf>

_____. **Why Google could die**. *Online*: FaberNovel, 2009. <http://www.fabernovel.com/en/analyze/news/why-could-google-die>

FOUCAULT, Michel. **L'archeologie du savoir**. Paris: Gallimard, 1969.

FREEMAN, Stephanie. The material and social dynamics of motivation: Contributions to Open Source technology development. *In*: **Science Studies**, n.2, v.20, 2007.

FREIBERGER, Paul e SWAINE, Micheal. **Fire in the Valley: The making of the personal computer**. McGraw-Hill, 2000.

FULLER, Matthew. **Software studies: A lexicon**. Cambridge: MIT Press, 2008.

GALISON, Peter Louis. **Image and Logic: A material culture of microphysics**. Chicago: University of Chicago Press, 1997.

GOOD, Byron. **Medecine, rationality, and experience: An anthropological perspective**. Cambridge: Cambridge University Press, 1994.

GRAHAM, Paul. **Hackers and Painters**. Sebastopol: O'Reilly, 2004.

_____. **The Power of a Marginal**. *Online*: ChangeThis, 6/09/2006. <http://www.changethis.com/26.03.PowerMarginal>.

GRIMMELMANN, James. Regulation by Software. *In*: **The Yale Law Journal**, n.7, v.114, 2005.

GROSS, Matthias. Productive Anarchy? Networks of Open Source Software development. *In*: **Forum: Qualitative Social Research**, n.1, v.8, 01/2007.

GUESSER, Adalto. **Software Livre e Controversias Tecnocientíficas**. Curitiba: Jurúa, 2006.

GUILLAUD, Hubert. **Yochai Benkler: Dépasser l'analyse de la topologie des reseaux**. *Online*: InternetActu, 02/06/2009. <http://www.internetactu.net/2009/06/02/yochai-benkler-depasser-lanalyse-de-la-topologie-des-reseaux/>

GUTERSON, Hugh. **Nuclear Rites**. Berkeley: University of California Press, 1996.

HANKS, William F. **Língua como prática social**. São Paulo: Cortez, 2008.

HANNEMYR, Gisle. **Technology and Pleasure: considering hacking constructive**. *Online*: FirstMonday, n.2, v.4, 1999. <http://131.193.153.231/www/issues/issue4_2/gisle/index.html>.

HEBDIGE, Dick. Posing … Threats, Striking … Poses. *In*: GELDER, Ken e THORTON, Sarah, **The Subculture Reader**, New York e Londres: Routeledge, 1997.

HELANDER, Nina e ANTIKAINEN, Maria. **Essays on OSS practices and sustainability**. Tampere, 2006.

HENKEL, Joachim, **Champions of Revealing - The role of Open source Developers in Commercial firms**. *Online*: Industrial and Corporate Change, 24/12/2008. <http://papers.ssrn.com/sol3/papers.cfm?abstract_id=946929>

HIMANEM, Pekka. **The Hacker Ethic and the spirit of the information age**. New York: Random House, 2001.

HOLTGREWE, Ursula e BRAND, Andreas. The polis of projects at work: Open Source Software development and the 'new spirit of capitalism'. *In*: **Osterreichische Zeitschrift fur Soziologie**, n.3, v.32, 2007.

IYER, Bala e DAVENPORT, Thomas. Reverse Engineering Google's Innovation Machine. *In*: **Harvard Business Review**, 04/2008.

JARDIM, José. **A construção do e-gov no Brasil: configurações político-informacionais**. *In*: Proceedings CINFORM - Encontro Nacional de Ciência da Informação V, Salvador de Bahia, 2004. <http://dici.ibict.br/archive/00000562/01/constru%C3%A7%C3%A3o_do_egov.pdf>

KAVADA, Anastasia. **Social Movements and current network research**. *In*: CAWN, Corfu, Greece, 6-7 novembro, 2003.

KELTY, Christopher. **Free Software/Free Science**. *Online*: FirstMonday, n.12, v.6, 2001. <http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/902/811>

_____. **Two Bits, The cultural significance of free software**. Durham e Londres: Duke University Press, 2008.

_____. Geeks, Social imaginaries, and Recursive Publics. *In*: **Cultural Anthropology**, n.2, v.20, 2005

KNUTH, Donald. **Literate programming**. *Online*: The computer journal, 1983 <http://www.literateprogramming.com/knuthweb.pdf>

KONZACK, Lars. **Geek Culture: The 3rd Counter-Culture**. *In*: FNG2006, 26-28/06/2006, Preston, Inglaterra. <http://www.vrmedialab.dk/~konzack/GeekCulture.pdf>.

KRISHNAMURTHY, Sand. **Cave or Community? An Empirical Examination of 100 Mature Open Source Projects**. *Online*: FirstMonday, n.6, v.7, 2002. <http://www.uic.edu/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1477/1392>

LAMMERS, Susan. **Programmers at work: interviews with 19 programmers who shaped the computer industry**. Tempus Books, 1989.

LANCASHIRE, David. 2005. **Code, Culture and Cash: The fading altruism of Open Source Development**. *Online*: FirstMonday, Special issue #2: Open Source, 2005. <http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1488/1403>

LATTERMANN, Chritoph e STIEGLITZ, Stefan. **Framework for governance in Open Source Communities**. *In*: 38th Hawai International Conference on System Sciences, 2005.

LESSIG, Lawrence. **Code and other laws of Cyberspace**. New York: Basic Books, 1999.

LEVY, Pierre. **De la programmation considérée comme un des beaux-arts**. Paris: La Découverte, 1992.

_____. **Qu'est-ce que le virtuel?**. Paris: La Découverte, 1998.

_____. **Cyberculture**. Paris: Odile Jacob, 1998.

_____. **La mutation inachevée de la sphère publique**. *Online*: IEML, 2008. <http://www.ieml.org/IMG/pdf/La_nouvelle_sphere_publique.pdf>

LEVY, Steven. **Hackers: Heroes of the comuter revolution**. New York: Anchor Press, 1984.

MACKENZIE, Adrian. The Performativity of Code: Software and cultures of circulation. *In*: **Theory, Culture & Society**, n.1, v.22, 2005.

MAHONEY, Micheal. Software: The self-programming machine. *In*: AKERA, Atsushi e NEBEKER, Frederik (org.). **From 0 to 1: An authoritative history of modern computing**. Oxford: Oxford University Press, 2002.

MARIS, Bernard. L'inventeur et le marchand. In: **Anti-Manuel d'économie** (Tome 2: Les cigales), Paris: Bréal, 2006.

MARKOFF, John. **What the dormouse said**. Londres: Penguin books, 2005.

MASSON, Matt. **The pirate's dilemma: How youth culture is reinventing capitalism**. Free Press, 2008.

MITNICK, Kevin e SIMON, William. **The art of deception: controlling the human element of security**. Indianapolis: Wiley Pub, 2002.

MOGLEN, Eben. **Anarchism Triumphant: Free Software and the Death of Copyright**. *Online*: FirstMonday.org, n.8, v.4, 08/1999. <http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/viewArticle/684>.

_____. **The dotCommunist Manifesto**. *Online*, 01/2003. <http://old.law.columbia.edu/publications/dcm.html>

MÜLLER, Martin. Open-Source – État des lieux. *In*: **Le Logiciel Libre**, Paris: O'Reilly, 2001.

NAGGUM, **Read the fine manual, please**. *Online*: USENET:comp.emacs, 14/11/1997. <http://groups.google.no/group/comp.emacs/browse_thread/thread/18de8730b68bea14/821a0f04bab91864>

NEGROPONTE, Nicholas, **Being Digital**. New York: Vintage, 1996.

NIEUWENHORF, Sashia van de. **Licensing Freedom: An Ethical Analysis of Free and Open Source Software Licenses**. Dissertaçao de mestrado, Utrecht University, 22/01/2008.

NITOT, Tristan. Entrevista no lemonde.fr. *Online*: **LeMonde.fr**, 30/05/2008.

NOISETTE, Thierry e PERLINE. **La Bataille du Logiciel Libre**. Paris: La Découverte, 2006 (2004).

OLIVA, Alexandre. Linux-libre e o dilema dos prisoneiros. *In*: **Linux Magazine**, n.54, 2009.

PAOLI, Stefano De e TELI, Maurizio. **Free and open sources licenses in community life: Two empirical cases**. *Online*: FirstMonday, n.10, v.13, 2008. <http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/2064/2030>

PERENS, Bruce. **The emerging economic paradigm of Open Source**. *Online*: FirstMonday, Special Issue #2: Open Source, 2005. <http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1470/1385>

PFAFFENBERG, Bryan. 'If I want it, it's OK': Usenet and the (outer) limits of Free Speech. *In*: **The Information Society**, n.365-366, v.12, 1996.

PINHEIRO, Alexandre Silva e CUKIERMAN, Henrique Luiz. **Free Software: Some brazilian translations**. *Online*: FirstMonday, n.11, v.9, 2004. <http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1189/1109>

PIRES, Hindenburgo Fransisco. Internet, Software Livre e Exclusão Digital: Impasses e opções no desenvolvimento de politicas publicas de alcance social no Brasil. *In*: **Geouerj**, n.12, 2002.

PRADES, Jacques. Community Development Corporations et Logiciels Libres. Une anthropologie comparée de formes coopératives. *In*: **Terminal**, n.80/81, 2004.

_____. Economie solidaire, technologies de l'information et territoire. *In*: **Terminal**, n.80/81, 2004.

RAYMOND, Éric. **The Cathedral and the Bazaar**, *Online*, 2000 (1997). <http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/cathedral-bazaar.ps>.

REIMENS, Patrice. Quelques reflexions sur la 'culture Hacker'. *In*: **Multitudes**, n.8, V.2, 03-04/2002.

ROBLES, Gregorio, GONZALEZ-BARAHONA, Jesus e MICHLMAYR, Martin. **Evolution of Volunteer Participation in Libre Software Projects: Evidence from Debian**. *In*: 1st International Conference on Open Source Software Systems, Genoa, 2005.

RONFELDT, D. e ARQUILLA, J. **Networks, netwars and the fight for the future**. *Online*: FirstMonday, n.6, v.10, 2001. <http://131.193.153.231/www/issues/issue6_10/ronfeldt/index.html>

_____. **The promise of nöopolitik**. *Online*: FirstMonday, n.8, v.12, 2007. <http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1971/1846>

ROSA, Guilherme. **Identidade cultural em comunidades de usuários e desenvolvedores de software livre: o caso Debian-RS**. Dissertaçao de Mestrado, PUC-RS, 2008.

ROSENBERG, Donald K. **Open Source: The unauthorized white papers**. Foster City, CA: M&T Books, 2000.

ROSENBERG, Scott. **Dreaming in Code**. New York: Three Rivers Press, 2007.

RUSHKOFF, Douglas. Open-Source Democracy. Online: Project Gutenberg, 2004. <http://www.gutenberg.org/etext/10753>

_____. **Evolution as a Team Sport**. *Online*: Ethical Technology (blog), 03/11/2005. <http://ieet.org/index.php/IEET/more/rushkoff20051103/>.

SANTOS, Adroaldo. **Inclusão Digital e desenvolvimento local no Brasil**. *In*: Congreso Internacional del CLAD sobre la Reforma del Estado y de la Administración Publica, Panama, 28-31/10/2003.

SAWYER, Keith. **Group Creativity: Music, Theater, Collaboration**. Mahwah, NJ: Lawrence Erlbaum Associates, 2003.

SCHNEIDER, Michel. **Ladroes de palavras**. Campinas: Editora da Unicamp, 1990.

SCHOONMAKER, Sara. Globalization from below: Free software and alternatives to neoliberalism. *In*: **Development and Change**, n.6, v.38, 2007.

SCHWARTAU, Winn. **Cybershock: surviving hackers, phrackers, identity thieves, internet terrorists, and weapons of mass disruption**. New York: Thunder's mouth press, 2000.

SHUTTLEWORTH, Mark. **Rethinking Gobuntu**. *Online*: gobuntu-devel@lists.ubuntu.com, 15/04/2008. <https://lists.ubuntu.com/archives/gobuntu-devel/2008-April/000650.html>

SILVEIRA, Sergio e CASSINO, Joao. **Software Livre e Inclusão Digital**. São Paulo: Conrad, 2003

SILVEIRA, Sergio. **Software Livre: A luta pela liberdade do conhecimento**. São Paulo: Fundação Perseu Abramo, 2004.

SLATALLA, Michelle e QUITTNER, Joshua. **Masters of Deception: The gang that ruled cyberspace**. New York: Harper Collins, 1995.

SOFTEX. **O Impacto do Software Livre e de Código Aberto na Industria de Software do Brasil**. Campinas: SOFTEX, 2005.

STALLMAN, Richard. **Free Software, Free Society**. Boston: GNU Press, 2002.

STEWART, Daniel. Social Status in an open-source community. *In*: **American Sociological Review**, n.21, v.70, 2005.

TAPSCOTT, Don e WILLIAMS, Anthony. **Wikinomics: How mass collaboration changes everything**. New York: Portfolio, 2008 (2006).

TOFFLER, Alvin. **A Empresa Flexível**. Rio de Janeiro: Record, 1985.

TORVALDS, Linus e DIAMOND, David. **Just for fun: The story of an accidental revolutionary**. New York: HarperCollins, 2001.

TUOMI, Ikka. Network of Innovation. **Change and meaning in the Age of the Internet**. Oxford: Oxford University Press, 2003.

TUKEY, John. The teaching of concrete mathematics. *In*: **American Mathematical Monthly**, 1958.

TURING, Alan. On computable numbers, with an application to the Entscheidungsproblem. *In*: **Proceedings of the London Mathematical Society**, n.46, 1936.

VIANNA, Tulio Lima. Por uma nova política de direitos autorais para a America Latina: O software livre como instrumento de efetivação do direito econômico ao desenvolvimento tecnológico. *In*: **Revista da Ordem dos Advogados**, n.1, v.6, 01/2006.

VON NEUMANN, John. **First draft of a report on the EDVAC**. 1945.

WARK, McKenzie. **A Hacker Manifesto**. Cambridge e Londres: Harvard University Press, 2004

WILLIS, Nathan. **The iPhone SDK and Free Software: not a match**. *Online*: Linux.com, 15/04/2008.
<http://www.linux.com/feature/131752>.