UNIVERSIDADE ESTADUAL DE CAMPINAS - **UNICAMP**
INSTITUTO DE FILOSOFIA E CIENCIAS HUMANAS - **IFCH**
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA POLÍTICA - **DCP**

UMA ANÁLISE SOCIOPOLÍTICA DO MOVIMENTO DO SOFTWARE LIVRE E DE CÓDIGO ABERTO

Antoine Bernard Marie Mazières
mazieres{at}lavabit{dot}com

Sob a orientação de **(Prof. Dr. Tom Dwyer)**

Dissertação apresentada como pré-requisito para obtenção do titulo de Mestre em Ciência Política, no programa de pós-graduação em Ciência Política da UNICAMP

Banca examinadora:	Prof. Dr. Tom Dwyer	DS-IFCH/UNICAMP
	Profa. Dra. Maria Conceição da Costa	DPCT-IG/UNICAMP
	Prof. Dr. Valeriano Mendes Ferreira Costa	DCP-IFCH/UNICAMP
(suplente)	Prof. Dr. Bruno Speck	DCP-IFCH/UNICAMP
(suplente)	Prof. Dr. Gilson Luiz de Oliveira Lima	IPA/CUM

Campinas_Setembro/2009.

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 morais e éticos. 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 – Moral and ethical 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

ANTOINE BERNARD MARIE MAZIERES

UMA ANALISE SOCIOPOLITICA DO MOVIMENTO DE SOFTWARE LIVRE E DE CODIGO ABERTO

Dissertação apresentada como pré-requisito para obtenção do título de Mestre em Ciência Política, no programa de pós-graduação em Ciência Política da UNICAMP

Aprovado em _____ de _____ de _____

BANCA EXAMINADORA:

_____

Orientador, Prof. Dr. Thomas Patrick Dwyer (DS-IFCH/UNICAMP)

_____

Prof. Dr. Maria Conceição da Costa (DPCT-IG/UNICAMP)

_____

Prof. Dr. Valeriano Mendes Ferreira Costa (DCP-IFCH/UNICAMP)

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.

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 *blackboxing* 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.

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.

AGRADECIMENTOS

Fabiano Mucillo, Tom Dwyer, Alain Alcouffe, Maria Conceição da Costa, Valeriano Mendes Ferreira Costa, Emerson Carvalho de Souza, Filomena Sandoval, Larissa Lisboa, Patrick Vezali, Gabriella Coleman, Paul O'Malley, Andreia Pio da Silva, Sydney Pio de Campos, Hélio Antonio Cassange Ortiz, Rose Meire Anésio Ferreira, Eduardo Piovani Dias, Silvio Jose Fernandes, Leandro Aparecido Roberto, Nicolas Cavayé, Samuel Huron, Jeremy Levy, Martine, Bernard, Philippe et Claire Mazières.

SUMÁRIO _

LISTA DE SIGLAS _

ADN - Ácido Desoxirribonucléico

API - *Application Programming Interface*
(Interface de programação de aplicativos)

APL - *Apache Public License*
(Licença pública de Apache)

BBC - *British Broadcasting Corporation*

BIND - *Berkeley Internet Name Domain*

BSD - *Berkeley Software Distribution*

CEGE - Comitê Executivo do Governo Eletrônico

DMCA - *Digital Millenium Copyright Act*
(Lei dos direitos autorais do milênio digital)

DNS - *Domain Name Server*
(Servidor de nomes de domínios)

DOS - *Disk Operating System*
(Sistema operacional em disco)

F/LOSS - *Free/Libre Open Source Software*

FOSS - *Free and Open Source Software*

FSF - *Free Software Fundation*
(Fundação para o software livre)

FTP - *File Transfer Protocol*
(Protocolo de transferência de arquivos)

gNS - gNewSense

GNU - *GNU is Not Unix*
(GNU não é Unix)

GPL - *General Public License*
(Licença pública geral)

GPV - *General Public Virus*
(Vírus público geral)

GSM - *Global System for Mobile communications*
(Sistema global para comunicação móveis)

HTML - *Hypertext Markup Language*
(Linguagem de marcação hipertexto)

IDABC - *Interoperable Delivery of European eGovernment Services*

IOSN - *International Open Source Network*

IP - *Internet Protocol*
(Protocolo de Internet)

iPDP - *iPhone Development Program*

IRC - *Internet Relay Chat*

JPEG - *Joint Photographic Experts Groups*

LGPL - *Lesser General Public License*

MIT - *Massachussetts Institute of Tecnhology*
(Instituto de tecnologia de Massachusetts)

MPEG - *Moving Picture Experts Group*

MPL - *Mozilla Public License*
(Licença pública de Mozilla)

MS - MicroSoft

NPL - *Netscape Public License*
(Licença Pública de Netscape)

ONU - Organização das Nações Unidas

OOo - OpenOffice.org

OSOR - *Open Source Observatory and Repository*

OSS - *Open Source Software*

PC - *Personal Computer*
(Computador pessoal)

PDA - *Personal Digital Assistants*
(Assistente pessoal digital)

PGP - *Pretty Good Privacy*
(Privacidade bastante boa)

SDK - *Software Development Kit*
(Kit de desenvolvimento de software)

SL/CA - Software Livre e de Código Aberto

SMTP - *Simple Mail Transfer Protocol*

SO (*OS*) - Sistema operacional (*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*
(Programa das nações unidas para o desenvolvimento)

VoIP - Voz sobre IP

XML - *eXtensible Markup Language*

LISTA DAS FÍGURAS _

(Capítulo I **)**

(Capítulo II **)**

(Capítulo III)

(Conclusão)

> ?

O melhor jeito de treinar [para se tornar bom programador] é de escrever programas, e de estudar excelentes programas que outras pessoas têm escritos... **Você deve querer ler o código das outras pessoas**, e depois escrever o seu, e depois chamar os outros para revisar seu código.

Bill Gates, 1989

Olhar código somente como "instruções para a máquina" é empobrecer uma perspectiva como olhar Shakespeare somente como "tinta sobre papel".

Samir Chopra e Scott Dexter, **2007**

Introdução

O movimento do software livre e de código aberto (SL/CA) é uma reação à privatização do código que aconteceu nos anos 1960. Desde então estruturou-se em vários ramos complexos e chegou a oferecer uma alternativa tecnológica concreta em várias áreas da sociedade de informação, principalmente no uso e a construção da internet.

Assim, várias mudanças esclarecem o sucesso do software livre, ou pelo menos sua atualidade nas políticas públicas, nos debates políticos e na literatura universitária. Primeiro, desde mais de uma década, o número de dispositivos informativos que acompanham o cotidiano dos indivíduos de classe média cresceu de maneira inédita. O celular e o computador pessoal são dois objetos comunicativos, cujas fronteiras afastam-se com a redução do consumo de energia e de tamanho junto com um melhor desempenho e estabilidade. Os recursos oferecidos tornam-se ferramentas básicas de muitas atividades que até então não aproveitavam dos efeitos de redes. Nesse contexto, o software livre encontra um sentido no senso comum como desafio técnico, permitindo a participação dos usuários além do que permite as plataformas proprietárias.

Ainda, uma das idéias que circula nos debates comuns sobre os programas de código aberto é a idéia de uma comunidade sem alfândega, cujo acesso é definido pela simples participação de quem a deseja. A imagem do usuário-desenvolvedor aproxima-se então da idéia de um cidadão que participa de um movimento coletivo: o movimento tecnológico. O "*You*" (Você), pessoa do ano em 2006 segundo o jornal TIME, representa bem essa corrente contemporânea que coloca o indivíduo como atuante decisivo da dinâmica social no mesmo tempo que o funde inteiramente a comunidade a qual ele participe. Nesse sentido, o movimento do software livre é um exemplo que dá muito sentido a essa "modernidade", mostrando realização como a Internet, possibilidades como a participação política direta, e perguntas como o modelo

econômico de apropriação dos saberes.

Enfim, por qualquer direção que for o movimento tecnológico atual, ele é influenciado por seus atuantes, e uma maioria de pessoa no mundo não tem acesso à rede e as ferramentas tecnológicas básicas. É nos países periféricos que a fratura digital é a mais aparente, as classes médias têm acesso às tecnologias de informação enquanto as classes baixas acumulam atraso na educação digital e não participam nem aproveitam dessas evoluções, embora elas pareçam estruturar profundamente a sociedade contemporânea. Nesse contexto, o software livre aparece como uma solução ótima para as políticas públicas de inclusão digital, sendo que permite oferecer tecnologias com custos reduzidos e potenciais de adaptabilidade maior.

Nesse trabalho, a análise foca-se sobre os aspectos culturais do movimento e principalmente seus determinantes políticos. Além disso, as comunidades "livres" apresentam-se sem discurso ou agenda político precisa, desenvolvendo então uma cultura independente como ramo da cultura "hacker". Para tanto, argumenta-se que embora esse "agnosticismo político" é característico do movimento SL/CA, ele é a face de uma prática social intensa, informal e normativa no contexto informacional, que constrói a aura política do movimento com um conjunto. Assim, a hipótese explorada nessa pesquisa é que a prática da programação entendida como pragmática, é o vetor constitutivo e determinante das práticas políticas informais e de suas conseqüências sociais. Assim, a seguir explora-se como esse estudo está sistematizado.

No primeiro capítulo, busca-se apresentar uma história breve da emergência, constituição e consolidação do movimento SL/CA no mundo. Voltada às origens do objeto software na história do início da computação, mostra-se ainda que as distinções "naturais" feitas entre software e hardware podiam ser interpretadas diferentemente antes que companhias, como a Microsoft ou a AT&T, venham isolar um objeto "software" e fechar seu código para disponibilizar um produto finito com opções de configuração e de apropriação limitada. Nesse contexto, o movimento SL/CA aparece como uma reação, ou seja, um esforço de manter uma tradição de livre interação no processo de inovação e de construir uma alternativa ao modelo proprietário que se fortalece com a produção em grande escala de computadores pessoais. Por dentro, o movimento tem estruturas comunitárias, jurídicas e institucionais complexas que tornam difícil delimitar uma linha de ação comum, além de querer promover a necessidade de um código fonte aberto para os programas. Contudo, a divisão institucional entre software "livre" e software "aberto", que acontece publicamente em 1998, permite entender melhor as lógicas econô-

micas e ideológicas atrás de diversos ramos do movimento global, e assim delimitar seus desafios contemporâneos.

Mais adiante, já no segundo capítulo, tenta reconstruir-se uma parte do esforço teórico feito pela literatura universitária em cima dos aspectos culturais do movimento SL/CA e da cultura hacker. Assim, a noção de ética hacker, entendida como construção social, permite isolar várias características dos atuantes do movimento do software livre. Porém, seguindo as reflexões da antropóloga Gabriella Coleman (2008) que afirma que o conceito de ética hacker fica ainda preso ao binário moral midiático, que se constituiu por volta da figura do hacker (Pirata ou Herói da era digital). Assim, há um intento para isolar o discurso político aparente como um agnosticismo político, cuja função é de permitir à paixão pela informação dos hackers de se exprimir livremente. Essa prática estrutura suas políticas informais, tecnológicas e práticas que tentaremos reduzir a uma tipologia de tipos ideais.

Ainda no segundo capítulo, apresenta-se o paradigma teórico de pragmática da programação; construído para interagir com a pesquisa técnica e bibliográfica. Aqui alertamos que o ato de escrever código acontece num contexto de padrões e expectativas que determinam sua estética. Isso acontece, comparando o fato de programar como o de realizar arte, lei, ou arquitetura física a partir de referências universitárias. Desse modo, mostra-se o código como um discurso normativo e político sobre a informação e seu tratamento pela máquina e a rede. De forma geral, a idéia é de mostrar que a pragmática da programação é o vetor do "agnosticismo político" do movimento pelo fato de ser o vetor das suas políticas informais, e que as lógicas do código que afastam as referências políticas tradicionais são as que articulam o discurso normativo do movimento sobre a informação.

No terceiro e último capítulo, propõe-se, então, uma tipologia de três pragmáticas emblemáticas de vários projetos open-source, que tem como propósito interagir com a tipologia das políticas informais do movimento SL/CA. Assim, as lógicas de transgressão, de tecnologia cívica, de inversão e de colaboração são reinterpretadas pelo vetor das pragmáticas de liberdade, abertura e segurança. Com análise de estudos de casos, referências técnicas e conceitos teóricos encontrados no esforço tecnológico do movimento do software livre, estabelece-se padrões de interpretações dos contextos político-tecnológicos pelos próprios atuantes e pelas comunidades.

Capítulo 1

Histórico e Definição

Segundo Paul Ceruzzi, há duas correntes históricas que se construíram nos arredores do objeto Software (CERUZZI, 1998). A primeira dessas correntes evita uma descrição demasiamente técnica por ser destinada a um publico amplo, interessa-se à formação das companhias privadas individuais (IBM, Apple, Microsoft). A segunda historiografia, por sua vez, foca-se sobre o nascimento das linguagens de programação (FORTRAN, COBOL, entre outros), questionando como cada linguagem apareceu e permitiu aos seus desenvolvedores de extrair novas funcionalidades do hardware.

Desde o trabalho de Ceruzzi, uma terceira corrente de historiografia foi se construindo paulatinamente nos arredores da observação do movimento do software livre e de código aberto durante os anos 80 e 90. Essa nova narrativa do movimento tecnológico computacional relatou as tradições da cultura hacker de programação. De suas origens nos laboratórios acadêmicos de informática dos anos 50 e 60, construiu-se a historia de como fatores econômicos e ideológicos formaram as tradições culturais e autorais da programação, e como essas recentes tradições afetaram e, até, determinaram que tipo de software esta sendo escrito.

Ademais, este capítulo introdutivo tem como propósito usar livremente essas três correntes de historiografia da computação para apresentar o movimento do software livre no seu contexto histórico, social e econômico. Num primeiro ponto, mostraremos como o conceito de software se formou como abstração do hardware e descreveremos os elementos concretos que recobrem, hoje, essas duas noções (1.1). Depois, mostraremos como o objeto software constitui-se em grande escala ao acompanhar a historia do computador pessoal e das companhias que o construíram. Assim, observam-se os primeiros

rastos da cultura *open-source*, como reação à privatização do código (1.2). Ainda, na terceira parte, mostraremos a constituição das comunidades e licenças que formam o movimento do software livre, como as instituições que o promovem. Assim, através dos conflitos de interpretação do que é ser "livre" ou "aberto" para uma tecnologia, observa-se a heterogeneidade do movimento que chama a postular de uma definição sistemática (1.3). Enfim, no quarto e ultimo ponto, relataremos alguns desafios contemporâneos que são em debates dentro das comunidades para ilustrar os sucessos e carências presentes no movimento (1.4).

1.1 Computador, hardware e software

1.1.1 A abstração do Software

A origem da palavra "computador" designa um homem realizando operações matemáticas com a ajuda de ferramentas mecânicas. Dentre as ferramentas que ficaram, as mais conhecidas são: o ábaco (2700-2300 a.C.), o mecanismo antikythera (150-100 a.C.) e, datando do Renascimento, a regra de cálculo e o astrolábio. Os primeiros rastros de programabilidade, ou melhor, de automação das operações de cálculos, datam também da antiguidade, com o teatro mecânico de Heron d'Alexandria (10-70 d.C.) encenando uma peça de 10 minutos, com a ajuda de objetos animados por um sistema de cordas e alavancas cuja execução estava decidida com antecedência. Associando essas primeiras ferramentas de automação do cálculo com a noção de programabilidade que podemos constituir a origem do sentido dado ao computador moderno. Assim, podemos falar do computador como uma máquina que trata dados (cálculos) a partir de instruções dadas (programabilidade). Em francês, o termo "computador"traduzia-se inicialmente por "calculador"(*calculateur*). Foi um professor de latim da Sorbonne, Jacques Perret quem, mandado pela IBM, batizou o objeto de *ordinateur* em referência à figura do ordenador na crença católica.

Entretanto, Foi somente entre 1940 e 1945 que surgiram os primeiros computadores semelhantes aos que são conhecidos atualmente. Tratavam-se de grandes máquinas ocupando quartos inteiros, consumindo a energia de centenas de micro-computadores semelhantes aos atuais. A revolução computacional dos anos 40 é o resultado da fusão entre vários esforços de exploração teóricos e práticos (BLACK, 2002). Baseando-se sobre a *tradição*

matemática constituídas pelas obras de Leibniz, Boole, entre outros, Alan Turing conseguiu expor as bases da computação de propósito geral nos anos 30, com a "Maquina Universal Turing". Isto é, uma representação abstrata e teórica de um dispositivo computacional. Segundo os comentários do próprio autor, "é possível inventar uma maquina única que poderia ser utilizada para computar qualquer seqüência"[1] (TURING, 1936, p.241).

Contudo, a inspiração abstrata de Turing realizou-se concretamente pelo meio de uma tradição distinta: a *tradição de engenharia*. Esse conjunto de escolas práticas permitiu, nos anos 40, que fossem construídas as primeiras maquinas eletrônicas de cálculos. Os primeiros dispositivos eletrônicos incorporando o sistema teórico de Turing foram o resultado da culminação das heranças empíricas de autores como John Napier (1550-1617), Blaise Pascal (1623-1666), Charles Babbage (1791-1871), entre outros.

Frente a essas maquinas, apareceu a necessidade de controlá-las, de produzir instruções que poderiam ser prescritas. Assim, como conseqüência dessa necessidade, nasceu o terceiro elemento que constituiu a computação moderna: *a tradição de programação computacional*. Isso aconteceu notavelmente no âmbito do esforço cientifico realizado durante a segunda guerra mundial, por exemplo, por volta da maquina ENIAC, desenvolvida pela Universidade de Pensilvânia.

Servindo os propósitos de guerra, as primeiras máquinas computacionais dedicavam-se à velocidade de calculo, e por isso, estavam construídas com propósitos particulares e isolados, de tal forma que se entendia as noções, hoje estabelecidas, de hardware e software, num objeto só, como fundamentalmente dependente, uma da outra. Contudo, a idéia de "propósito geral" que se encontra no sistema de Turing, designa por tanto uma plataforma compatível com qualquer conjunto de instruções, como um dispositivo genérico capaz de realizar cálculos variados e adaptáveis. Em outros termos, embora as raízes teóricas da computação tivessem diferenciado hardware e software, as primeiras realizações práticas criaram maquinas que não permitiam a esses dois objetos de evoluir independentemente. Essas limitações apareceram com nitidez ao estender o uso dessas primeiras maquinas no mundo corporativo. Segundo as palavras de Maurice Black: "A chave para construir um computador usável – e conseqüentemente profitável – não é o fato de concebê-lo para desempenhar funções ou cálculos específicos, mas o

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

de construir um hardware de propósito geral que poderia ser programado facilmente para qualquer propósito."[2] (BLACK, 2002, p.40). Porém, como nota o historiador da computação Micheal Mahoney sobre esse período:

> Nos temos praticamente nenhum relatório histórico de como, começando nos anos 1950, governos, negócios, e indústrias, colocaram as suas operações no computador. Fora alguns estudos com um foco sociológico sobre os anos 70, a programação como nova atividade técnica, e os programadores como nova força de trabalho, não receberam atenção histórica.[3]
>
> MAHONEY, 2002, p.92.

Contudo, embora seja difícil dar atenção aos esforços isolados que permitiram a abstração progressiva do software como um objeto próprio, sujeito a evoluções independente da sua plataforma de execução, podemos sublinhar aqui um dos fatos importante da historia da computação, que permitiu que emergisse o conceito moderno de software. No âmbito das problemáticas procurando construir um hardware de propósito geral, o matemático John Von Neumann descreveu (VON NEUMANN, 1945) uma arquitetura cuja pretensão era de ser inteiramente digital, isto é, que as instruções sejam armanezadas na memória do computador – e não como circuito mecânico específico. Assim as instruções podiam ser transportadas de um computador ao outro. Esta idéia de "programa armanezado"mudou a face da computação e diferenciou o objeto software, do hardware.

Por um lado, isso permitiu às instruções serem desenvolvidas abstratamente, sem laço físico com o material da maquina, e por isso abriu as possibilidades de um desenvolvimento tecnológico colaborativo. Por outro lado, isto foi a origem do fenômeno de *blackboxing* (caixa preta) dos programas, permitindo a seus desenvolvedores de entregar um produto finito cujo código fonte podia ser escondido. Assim, por essas duas razoes principais, Black

[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."

[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."

pôde afirmar que “a abstração do software do hardware é o aspecto o mais importante do inicio da historia da programação” (BLACK, 2002, p.49).

Portanto, o conceito de software levou tempo a ser usado pelos atuantes das tecnologias informáticas. Segundo Matthew Fuller, o primeiro uso da palavra que foi publicado acha-se num artigo da revista *American Mathematical Monthy* de 1958 (TUKEY, 1958, citado em FULLER, 2008). Tratava-se então da idéia que todos os problemas matemáticos podiam ser resolvidos por dentro das matemáticas, idéia que se encontra hoje mais por volta do conceito de algoritmo, entendido como abstração livre dos detalhes de implementação. Porém, é em 1968, quando IBM decidiu de dividir sua secção informática em dois departamentos, hardware e software, para escapar de um processo de Antitrust da administração americana, que o termo "software"levou boa parte de seu sentido comum. Contudo, será com a propagação dos computadores pessoais e a democratização do acesso às tecnologias computacionais, que o objeto "software"aparecera com toda sua dimensão social e política.

1.1.2 Elementos principais do hardware e software

Embora sua estrutura tenha evoluído com o tempo, o computador é geralmente composto dos mesmos tipos de elementos: uma unidade de controle central que gere os diferentes componentes, uma unidade aritmética e lógica que realiza as operações e uma unidade de memória na qual os dados são armazenados e lidos. O computador com um todo recebe entradas (*input*) e restitua saídas (*output*). Essa dupla *input*/*output* pode ser reproduzida a todos os níveis da hierarquia que compõe um computador: entrada e saída para um programa, um componente, um circuito, etc. O conjunto de elementos físicos de um computador constitui seu hardware. A sua fabricação foi modificada em função de objetivos de rentabilidade, técnicos, espaciais que transformaram suas raízes mecânicas (transistores, tubos, entre outros) em circuitos miniaturas integrados. Vale mencionar que, as últimas pesquisas teóricas a respeito de hardware apontam tecnologias quânticas, químicas e ópticas que indicam desempenhos sem medida com as tecnologias atuais das mais avançadas.

Esse conjunto material gera dados, programas, protocolos, ou seja, um conjunto imaterial designado pelo termo genérico de software. Todos os softwares são criados com uma ou várias linguagens de programação. Trata-se de linguagens exclusivamente escritas, construídas para serem precisas e concisas, evitando assim possíveis ambigüidades. Uma vez que é interpretado

pelo hardware (compilado), o software somente aparece como uma sucessão de bytes (zero e um), um fluxo binário incompreensível pelo ser humano (Figura 1.1). Embora os softwares possam designar um número ilimitado de objetos, podem ser classificados em vários tipos, vejamos:

- O *Sistema Operacional* (SO), como por exemplo, Windows, Unix ou DOS. Ele organiza e coordena a atividade e a distribuição dos recursos finitos do hardware.
- As *bibliotecas*. Um conjunto de "sub-rotinas"chamadas para desenvolver programas maiores.
- Os *dados*. Trata-se tanto de protocolos de transferência (SMTP, FTP) como dos formatos de arquivos (JPEG, HTML, MPEG, XML).
- Os *aplicativos*. Podem ser ferramentas para escritórios (MS Office, OpenOffice.org), internet (browser, servidores web), gráficos (editor de imagens, criação 3D), áudio (Composição musical, mixagem), engenharia do software (compilador, debugger), jogos interativos, entre outros.

Figura 1.1: Código fonte e formato binario

`$ echo ‘‘Hello World’’`	`011001010110001101101000011011110010000000100010010010000110010101101100011000110111100100000101011101101111011100100110110001100100000100010`

```
$ echo ‘‘Hello World’’          01100101011000110110100000
                                11011110010000000100010010
                                00100000110010101101100011
                                01100011011110010000000101
                                01110110111101110010011010
                                1000110010000100010
```

1.2 P.C. e Software

1.2.1 O P.C. e a informática de massa

Os computadores custavam tão caros para serem comprados, usados e mantidos que somente algumas universidades, empresas e órgãos governamentais

os possuíam. Graças às evoluções tecnológicas que reduziram custos de produção, o espaço e a energia utilizadas por cada unidade, surgiram nos anos 70 os primeiros micro-computadores, ou computadores pessoais (*Personnal Computer*, PC), que o termo "computador"vem designar hoje por padrão. Designando unicamente computadores de escritório (*Desktop*), recentemente esse termo designa também os computadores portáveis (laptop, notebook) e os dispositivos reduzidos (mini-notebook, PDA, etc). Estes são geralmente utilizados por uma pessoa sozinha, o que os diferenciam dos servidores.

Com conseqüência disso, há um ambiente específico nos arredores dessas inovações tecnológicas. No período de revolução cultural americana dos anos 70, os campi universitários (particularmente o da Universidade de Stanford, na Califórnia) eram os lugares de encontro dos grandes nomes da produção de PCs, dentro dos quais se encontra Steve Jobs e Steve Wozniac (co-fundadores de Apple), Bill Gates e Paul Allen (Microsoft), Paul Graham, Bill Joy e outros. Eles conviveram com diversos movimentos sociais: marxistas, contra a guerra no Vietnam, zen-budista, ecologista, rock, ficção científica, entre outros (BRETON, 1990). Se esses movimentos sociais são relativamente ignorados por esses recentes atuantes das novas tecnologias, eles ficaram influenciados por um ambiente de hyper-diversidade alternativa, de transgressão e de inovação que formará as bases do que tornar-se-ia "os piratas do Vale do Silício"[4].

De acordo com Paul Graham, engenheiro da computação e contemporâneo dessa época, testemunha no seu ensaio *The Power of a Marginal*: "O mundo ainda não tinha consciência que o fato de iniciar uma companhia de computadores era da mesma ordem do que ser artista ou pintor" (GRAHAM, 2006). Assim, Aceitando a idéia que as coisas novas e brilhantes são inspiradas e crescem às margens de uma sociedade, foi a coexistência aberta de muitas margens culturais num mesmo tempo e espaço que permitiu um ambiente de liberdade e audácia científica auspicioso às revoluções tecnológicas.

Os criadores dos computadores pessoais não imaginavam o sucesso que teria seu modelo tecnológico (NEGROPONTE, 1996). Eles aproveitaram dessa situação usando inovações a custos reduzidos. Por exemplo, a companhia Xerox que inventou a interface gráfica e o mouse, partilhou suas inovações com Apple, sem receber verba nenhuma. Os chefes da Xerox não acreditavam na utilidade dessas ferramentas e permitiram à Apple ver, informalmente,

[4]Título do filme de Martin Burke (1999) inspirado pelo livro: FREIBERG, Paul e SWAINE, Micheal, *Fire In The Valley*, 2000.

mas em detalhe, como funcionavam. Contudo, essas invenções contribuíram bastante ao sucesso do *Apple II* em 1977 e o do PC de maneira geral.

Finalizando, as primeiras produções de computadores pessoais foram realizadas por grupos reduzidos em condições precárias, o que não impediu alguns de transformar suas estruturas e acompanhar o alargamento do mercado de consumidores, como é o caso para Apple e Microsoft. Porém os grandes grupos, como IBM ou HP, não demoraram a entrar no mercado e propor produtos equivalentes. Além de se estender a um público de engenheiros, o PC apareceu como um meio de trazer uma melhor qualidade de trabalho (TOFFLER, 1985), e foi nos escritórios que o PC foi bem aceito em grande escala, mais rapidamente. O modelo de estação de trabalho (*Workstation*), considerado como uma plataforma de alta produtividade, substituiu paulatinamente as ferramentas do escritório tradicional.

1.2.2 O desenvolvimento dos Softwares e o início da cultura open-source

O software e o conjunto de aplicativos que compõem um sistema não são o foco principal da indústria da informática. Exceto a Microsoft, nenhum atuante desenvolve um sistema operacional com pretensão universal, ou seja, aspirando a funcionar em cima de qualquer hardware. Como confiou o P.D.G. da Apple a respeito de Bill Gates, durante uma entrevista no show *All Things Digital*: "Ele criou a primeira companhia de software antes que alguém nessa indústria saiba o que é uma companhia de software"[5]. De fato, nessa época a indústria de informática se preocupava principalmente em vender e manter o material. Os softwares eram considerados acessórios porque a maioria dos utilizadores os desenvolviam por conta própria. Foi a aparição de um sistema com tempo partilhado (*multitask*) que impôs a forma dos softwares que conhecemos hoje, tornado possível pela comercialização de disco rígido e de banda magnética permitindo armazenar, modificar e reutilizar programas (MÜLLER, 2001).

Uma informação importante é que numerosos sistemas compartilhados que rodavam nos computadores centrais de grandes instituições utilizavam a tecnologia Unix. Esta tecnologia foi desenvolvida por Ken Thompson e Dennis Ritchie dentro dos laboratórios da AT&T (Bell), a primeira versão desse sistema operacional aparece em 1969. Entretanto, por causa de uma

[5] *All Things Digital*, 30 de maio 2007.

ação anterior na justiça (1949-1956) por abuso de posição dominante, e para evitar qualquer infração ao julgamento, Unix não foi comercializado. O programa foi colocado sob licença que as instituições conseguiriam comprar. O software sendo distribuído sem serviço pós-venda nem correção de bug, constituiu-se um esforço ativo de desenvolvimento nas universidades. A rede *Usenet* tornou-se uma plataforma de troca de informações e de suporte para a utilização de Unix, e as inovações e correções de bugs circulam rapidamente. Além de ter um papel de coordenador, a Universidade de Stanford (Berkeley, CA-EUA) começou a desenvolver sua própria versão de Unix – BSD (*Berkeley Software Distribution*) – publicada pela primeira vez em 1978 por Bill Joy. Quatro anos depois, em 1982, outras versões de Unix foram publicadas de forma comercial por IBM, HP, DEC, para executarem-se no hardware existente. Na mesma data Bill Joy saiu da Universidade de Berkeley e fundou a SUN cujas máquinas usaram BSD 4.2. Em 1984, AT&T conseguiu, após uma nova ação em justiça, entrar no mercado da informática como um ator próprio e publica pela primeira vez uma versão comercial de Unix cujo código fonte ficara fechado dai em diante (MÜLLER, 2001).

Com isso, entendemos que Unix é a primeira proposta suficientemente realizada que permitiu pensar-se em um sistema universal. Com um código acessível, desde sua origem, aproveitou aperfeiçoamentos, modificações e adaptações que permitiram uma apropriação estendida, variada e criativa. Além disso, a influência desse sistema operacional ultrapassou os construtores de servidores. De fato, o ancestral da Internet, Arpanet, foi organizado e instalado a partir de sistemas Unix. Essa Tecnologia, embora esteja proprietária, foi apropriada por toda uma geração de engenheiros numa dinâmica de troca, de divisão e de aperfeiçoamentos constantes. É provavelmente aqui que se acham as raízes da filosofia *open-source*, no meio do ardor das revoluções culturais e tecnológicas dos anos 70 nos Estados-Unidos. Parece que a interdição para AT&T comercializar seu produto, permitiu que um novo modelo de desenvolvimento emergisse no mundo do software, deixando um espaço vazio no processo de radicalização proprietária da indústria da informática e particularmente do mercado imenso que abrir-se-ia com o computador pessoal.

1.3 Software Livre e de Código Aberto (SL/CA): comunidades, licenças e instituições

1.3.1 Inventar e proteger um software livre

Foi uma pequena falha no sistema de proteção do produto Unix que permitiu em um curto prazo, que ele fosse apropriado, desenvolvido e aperfeiçoado por um conjunto de atuantes de diversos locais. Dessa curta fase ficou uma tradição de troca "aberta"de informações e, embora já estivesse presente no meio da informática, tratava-se, para seus adeptos, de protegê-la para que seus benefícios não sejam apropriados. A aposta foi rapidamente entendida e desde a primeira versão de Unix desenvolvida pela Universidade de Berkeley, uma licença foi criada. A licença BSD obriga os desenvolvedores e usuários a mencionar os nomes dos autores do dito programa. Ela exclui toda responsabilidade dos programadores em caso de disfunção e o código é aberto, mas pode ser fechado, por exemplo, para uma utilização comercial.

Além disso, A *Free Software Fondation* (FSF – Fundação para o Software Livre), fundada em 1984 por Richard M. Stallman, vai articular, em termos ideológicos, e políticos os desafios de um software para ele ser chamado de "livre". A idéia é que cada linha de código é uma parcela de informação que, sendo traçada, partilhada e discutida, adaptar-se-ia melhor. Podemos observar que as metáforas não faltam ao discurso da fundação, as receitas de cozinha, por exemplo: descobrindo-se que batendo ovos e jogando-os numa estufa se faz uma omelete; porque deveríamos limitar nossa liberdade de partilhar essa dica com nosso vizinho, e nossa curiosidade de desenvolver outras receitas derivadas? Com essa pergunta simples, Stallman não defende um software de graça, como pode ser entendido na palavra inglês *free* ("*free as free beer*"), mas um modelo livre de desenvolvimento ("*free as free society*"). Por isso, se chama livre e não gratuito em português. Também, é por isso que a apelação a mais utilizada pelas instituições internacionais, adicionam à sigla inglesa, o termo *libre* da língua francesa e espanhola: *Free/Libre and Open-Source Software* (F/LOSS).

Seguindo esse propósito, foi criado a Licença Pública Geral (GPL – *General Public License*) que inspirou o movimento do software livre. Um código sob licença GPL é aberto e modificável, e deve ficar desse jeito. Nenhuma restrição às condições de base pode ser adicionada, e em complemento ao seu interesse jurídico, a GPL veicula uma ideologia simbolizada pelas "quatros

liberdades"[6]: a liberdade de executar o programa, para qualquer propósito; a liberdade de estudar como o programa funciona e de puder adaptá-lo a suas necessidades; a liberdade de redistribuí-lo; a liberdade de melhorá-lo e partilhar com a comunidade.

Protegida por essa licença, a FSF iniciou um projeto ambicioso, aspirando oferecer um sistema operacional e aplicativos com bom desempenho e inteiramente livres. Então, no quadro do projeto GNU (acrônimo de *GNU is not Unix*), centenas de programadores das regiões desenvolvidas do mundo acabaram desenvolvendo aplicativos semelhantes aos oferecidos pelo sistema Unix. Vários aplicativos criados nessa época ficaram até hoje conhecidos por ter bom desempenho (notavelmente GCC). Apesar de vários anos de trabalho, ainda não estava disponível um núcleo (*kernel*) que permita o funcionamento de todos os aplicativos em um sistema único. Em 1991, Linus Torvalds, programador finlandês, publicou o código de seu *kernel* "Linux" que juntava as realizações até então feitas, num sistema só, independente do *kernel* proprietário da AT&T. Desse jeito, completava-se os primeiros objetivos do projeto GNU e realizava-se o primeiro sistema operacional universal sob licença GPL: GNU/Linux.

1.3.2 Software Livre *e/ou* de Código Aberto

Enquanto o movimento do Software Livre evoluiu num ambiente ainda pouco conhecido para a época, reivindicações cresceram internamente que apontavam a necessidade de abrir-se a patrocínios de instituições privadas. De fato, varias comunidades reclamaram que o sucesso do Linux não era ligado a sua característica de "livre"mas às suas vantagens e estabilidade oferecidas pelo acesso livre ao código fonte e seu modelo de desenvolvimento colaborativo. O ano de 1998 ocorreu uma cisão institucional para o movimento do software livre. Depois de 7 anos de vida, o sistema Linux não era mais marginal e começava a aparecer como uma alternativa credível para as empresas. Além da junção da Intel a *Linux International*, a IBM, por sua vez, construiu um hardware para rodar especificamente com o Linux, a empresa Netscape públicou o código fonte de seu navegador no site Mozilla.org. Ademais, os códigos de StarOffice (predecessor de OpenOffice) e de Solaris (Sistema Operacional da Sun) foram publicados. Muitas vozes se juntam por volta de eventos como a

[6]Referência às four freedom proclamadas pelo Presidente dos EUA, Franklin D. Roosvelt durante o tradicional Discurso Anual ao Congresso do dia 06/01/1941.

publicação do ensaio de Eric Raymond, *The Cathedral and the Bazaar* (RAYMOND, 1997), e o *Open-Source Summit*, realizado pelas edições *O'Reilly & Associates*. A idéia desenvolvida nesses eventos é de afastar as pretensões ideológicas da FSF e a defesa de um direito humano a co-criar/modificar a produção de um código e assim privilegiar a potencia e a rentabilidade de um modo de desenvolvimento. Foi no âmbito dessas idéias que se construiu o movimento *open-source* (de código aberto) por parte das instituições privadas (IBM, Intel, O'Reilly, SUN, RedHat, etc.) e de uma filosofia pragmática dos princípios que fizeram o movimento do software livre.

Essa denominação de *open-source* insiste sobre o código aberto e coloca de lado os compromissos políticos e ideológicos que ficarão, daqui por diante, reivindicações quase-exclusivas da FSF. Agrupam-se geralmente com o nome genérico de *Free/Libre and Open Source Software* (FOSS, F/LOSS) e em português: Software Livre e de Código Aberto (SL/CA). Para seus atores, os nomes de "movimento *open-source*" ou de "movimento do software livre" convocam realidades e ambições bem diferentes que recusam ser assimiladas umas – ideológicas – com as outras – pragmáticas. Porém, se os defensores do *open-source* fazem livremente referência aos ícones da asa livre do movimento, o inverso não é bem-vindo e, a simples pronunciação de "*open-source*"nos canais IRC do projeto GNU ganha um "não se faz open-source aqui!".

Figura 1.2: Apelações do movimento do software livre.

SL/CA, FOSS, F/LOSS

Asa ideológico-política	Asa pragmática
Free Software, Software Livre	Software de Código Aberto *Open-Source Software (OSS)*
Exemplo de comunidades: GNU	Exemplo de comunidades: BSD, Mozilla

1.3.3 Licenças e instituições

Desde o início do movimento FOSS, e mais ainda desde a cisão explícita de 1998, multiplicaram-se as licenças para proteger o trabalho das comunidades. Espalharam-se muitas licenças, cuja semelhança é a proteção numa certa medida da acessibilidade ao código fonte, porém se diferenciam pelas restrições

que trazem às quatro liberdades fundamentais do movimento do livre. Se alguns dos projetos financiados por empresas ficam sob GPL, muitas instituições, tanto privadas como públicas, preferiram desenvolver licenças próprias, afastando-se, por vezes, dos critérios de "liberdades" da FSF. Com freqüência, licenças específicas foram criadas para um projeto único; contam-se centenas delas cujas diferenças são a utilização com um software proprietário, o acesso às modificações, a publicidade, os direitos particulares do dono do projeto. A seguir, uma tabela de comparação de algumas famosas licenças, segundo os critérios descritos acima[7]:

Figura 1.3: Comparação de algumas das principais licenças "livres".

Licença	Utilização com software comercial	Acesso para todos às modificações	Publicável sob certas condições	Presença de direitos particulares reservados ao dono da licença
GPL (*General Public License*)	Não	Sim	Não	Não
LGPL (*Lesser General Public License*)	Sim	Sim	Não	Não
BSD (*Berkeley Software Distribution*)	Sim	Não	Não	Não
NPL (*Netscape Public License*)	Sim	Não	Não	Sim
Domínio Público	Sim	Não	Sim	Não

De posse dessa informação, sabemos que certas licenças criadas para proteger um projeto específico são reutilizadas para projetos independentes. De fato, alguns projetos de código aberto têm exito e chegam a ter uma influência própria dentro do mundo do "livre". Assim, o projeto Apache, com mais de 50 % do mercado de servidor web, criou uma licença própria (*Apache*

[7]Tabela copiada do livro coletivo, *Tribune Libre: Ténors de l'Informatrique Libre*, editora O'Reilly, p.200.

Public License), que foi copiada por muitos outros projetos. Essa licença é compatível com a GPL desde sua versão 2.0 (janeiro 2004) e é usada notavelmente por Google para alguns projetos abertos. A fundação Mozilla, dona do navegador Firefox, propõe uma licença incompatível com a GPL (*Mozilla Public License*) e apesar disso recuperada pela SUN (*SUN Public License*).

Isso pode representar uma concorrência para o mundo do desenvolvimento de software e sua dinâmica. A idéia da GPL foi de permitir a um conjunto de códigos em ser compiláveis, juntáveis, integráveis, assimiláveis. Por um lado, o fato de o código ser protegido pela GPL faz que ele só seja reproduzido/modificado sob o mesmo regulamento. Por isso, alguns chamaram a GPL de Vírus (GPV – *General Public Virus*)[8] criando e espalhando um único monopólio. Por outro, com o espalhamento de licenças livre distintas, somos confrontados a novos problemas jurídicos, vejamos:

> Qual estatuto deve adotar um software contendo códigos de licenças diferentes – por exemplo, um programa que integra ao mesmo tempo código MacOS da Apple, X Server e Mozilla para dar uma versão livre de Netscape Communicator? se Além disso, a camada usuário desse novo programa fictício usa uma biblioteca Qt da Troll Tech, uma terceira licença entraria no jogo. As três provocariam limitações diferentes.[9]
>
> MÜLLER, 2001, p.16.

Podemos dizer que, a área de desenvolvimento de software acha-se desacelerada pela definição de estratégias jurídicas e indústrias caras e freqüentemente laboriosas. Além das diferenças sobre os efeitos externos jurídicos do modelo aberto/livre, encontramos um processo igualmente confuso na escada institucional dessas definições jurídicas. De fato, há uma concorrência no que se respeita às classificações das licenças, como meio de concorrência de legitimidade para poder definir o que é "livre/aberto" e o que não é. A

[8] Expressão pejorativa comum dentro das primeiras criticas feitas à (L)GPL.

[9] "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 entrainent des limitations différentes."

FSF mantém um repertorio atualizado[10] de todas as licenças que se apresentam como livre e as classificam em função de suas compatibilidades com as diferentes versões da GPL, ou seja, sua própria definição do livre. A *Open Source Initiative*, fundada por Eric Raymond em 1998, então apoiada pela empresa Netscape, mantém um banco de dados próprio de licenças[11], cada uma sendo submetida a um processo de aprovação respondendo a uma "definição do Open-Source", sublinhando abertura e modelo de negócio. Da mesma maneira, a ONU, pelo intermediário de seus programas de desenvolvimento (UNDP – *United Nations Developement Programs*), fundou uma instituição com o propósito de definir e desenvolver o software livre e de código aberto.

Assim, o *International Open-Source Network* (IOSN)[12], que age particularmente na região asia-pacífica, também define e classifica os projetos livres e suas licenças em benefício de uma concepção híbrida (Liberdade/modelo de negocio), voltado às problemáticas de inclusão/exclusão digital e de governo eletrônico. Os governos nacionais, particularmente os da França, do Brasil, da China e de Israel, focam-se mais sobre os casos de adoção pela administração de sistema open-source, favorecendo critérios de gratuidade (redução de custo) e de acesso ao código fonte (controle/segurança). Na Europa, é o caso da *Interoperable Delivery of European eGovernment Services* (IDABC) e do *Open Source Observatory and Repository* (OSOR). No Brasil, iniciativas semelhantes eram quase inexistentes antes de 2000 e ficavam somente em nível local, como é o caso dos telecentros que privilegiaram pouco a pouco o uso do Linux. O software Livre tornou-se prioridade no nível federal após um decreto presidencial de 2003[13] fornecendo ao CEGE (Comitê Executivo do Governo Eletrônico[14]) a "implantação do software livre". Sendo assim, o software livre é considerado como recurso estratégico, uma opção tecnológica favorecendo, a curto e longo prazo, uma autonomia e uma apropriação independente das ferramentas tecnológicas, usadas de maneira transversal em todas as realizações técnicas do governo eletrônico. Segundo Vianna (VI-

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

[11]http://www.opensource.org/licenses

[12]http://www.iosn.net. Para os desenvolvimentos a respeitos de licenciamento de projetos livre, ver: http://www.iosn.net/licensing/foss-licensing-primer/

[13]Decreto presidencial do dia 29 de dezembro de 2003

[14]O Comitê Executivo do Governo Eletrônico (CEGE) foi criado em 18 de outubro de 2000 no âmbito do Conselho de Governo da Presidência da República, com o objetivo de "formular políticas, estabelecer diretrizes, coordenar e articular as ações de implantação do Governo Eletrônico".

ANNA, 2006), o software livre é um meio que garante concretamente o direito econômico ao desenvolvimento tecnológico. Esse direito traz a idéia de uma autonomia cientifica nacional como a de democratização do acesso as novas tecnologias. Seria, de acordo com as palavras do autor, uma arma "contra a colonização tecnológica dos Estados-Unidos" e um meio de garantir uma formação mais independente das elites.

1.3.4 Postulado de definição

Vejamos, então, que comunidades, instituições e licenças se opõem ao definir os propósitos e meios do chamado 'movimento do software livre' e assim criam uma concorrência para conquistar a legitimidade em definir o movimento. Para o sociólogo, trata-se de conseguir capturar uma forma sistemática de um objeto cuja natureza é fluida e evolutiva, como é freqüentemente o caso a respeito dos movimentos sociais, por exemplo, nas analises do movimento alter-globalização (KAVADA, 2003).

No nosso caso, as analises de rede propostas por Arturo Escobar (ESCOBAR, 2003) parecem fazer sentido com a estrutura de baixo para cima (*bottom-up system*) e o alto grau de auto-gestão que se encontra nos arredores do Software Livre e de suas comunidades. Considerando a metáfora da "malha"(*meshwork*) em vez da de "rede"(*network*), o autor permite a observação de uma grande heterogeneidade dentro de um mesmo movimento social.

Desse jeito, podemos incluir todas as contradições sublinhadas pelos diferentes atuantes do movimento. Levando três exemplos emblemáticos, os das comunidades BSD, GNU, Mozilla, observamos que a primeira -BSD- não é ligada a uma instituição, mas a uma licença e permite usos comerciais que fechem o código. Por isso ela esta excluída da visão do "livre"da comunidade GNU, a qual esta ligada a uma instituição (FSF) e uma licença (GPL). Os softwares GNU são reconhecidos como "livres"pelas duas outras comunidades comentadas aqui. Por sua vez, a terceira comunidade -Mozilla- é também excluída pelos critérios da FSF, embora não permite o encerramento do código de seus produtos. Do ponto de vista de GNU, só GNU é "livre". Do ponto de vista de BSD, todas três são "livres". Do ponto de vista de Mozilla, só Mozilla e GNU são "livres". Ademais, a respeito da apelação "*open-source*": GNU não quer se considerar Open-Source, enquanto BSD como Mozilla, consideram as três comunidades como Open-source.

Neste âmbito de oposições, postular que o movimento do livre é composto

de todos os atores que pretendem fazer parte do mundo do "livre"ou "*open-source*"parece um meio eficiente para capturar o movimento com toda sua complexidade, sendo possível de revelar as nuances identitárias dos atuantes ao observar um contexto determinado.

1.4 O sucesso do código aberto e seus desafios contemporâneos

A mudança que aconteceu em 1998 com a cisma entre os mundos do "livre"e do "open source"tem como maior conseqüência a entrada das empresas e a atribuição de orçamentos, até então inesperados. Tais atitudes são importantes meios que são estabelecidos por empresas como IBM ou SUN para adaptarem os softwares as suas necessidades. É nessa época que vão se formar umas das vitorias decisivas do "livre"para sua fama. Por exemplo, em 1998, a IBM entra no projeto Apache, um servidor Web, mostrando uma estabilidade notável em seu desenvolvimento, enquanto os equivalentes proprietários se perdiam em reorientações constantes. Hoje, o Apache é umas das conquistas significativas do mundo do software livre, hospedando a maioria dos servidores Web do mundo (70%)[15]. A IBM investiu nessa época um milhão de dólares no Linux para colocá-lo nos servidores dele. O sistema operacional FreeBSD aproveitou dos investimentos de Apple e Google. Outros projetos conhecem um sucesso notável: DNS e BIND, SendMail, Samba, Perl, Python, Tcl/Tk.

Podemos observar que as aplicações livres que ganharam sucesso são relativamente "fundamentais", ou seja, sistemas e programas para servidores, protocolos de redes ou de arquivos, linguagens de programação. Neste limbo, o mundo do Desktop, do computador de escritório, do usuário final (*end-user*), está para ser conquistado. Basicamente, uma máquina desktop é composta por um sistema operacional no qual são adicionados alguns elementos essenciais. Com 90% do marketshare[16], a Microsoft tem quase o monopólio do sistema operacional utilizados pelos computadores pessoais (*Microsoft Windows*) e reduz o uso do Linux a uma parte mínima de usuários avançados (0,90%). Todavia, desde 2005, Mark Shuttleworth, financia, através da sociedade de serviço *Canonical Ltd.* (Reino-Unido), uma versão de Linux

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

(Ubuntu) com objetivo de ser intuitiva e dedicada aos usuários comuns. Essa distribuição do Linux parece conseguir seu caminho no mundo do desktop, notavelmente com a sua adoção por grandes instituições (Wikimedia, Amazon.com, Policia Militar Francesa). O OpenOffice.org (OOo), alternativa ao Microsoft Office, é um projeto já antigo que se beneficia do apoio da SUN e acha-se por padrão em quase todas as distribuições do Linux, e fica também disponível para os usuários de Windows e Apple. Uma das maiores conquistas do "livre"no mundo do desktop é o browser Mozilla Firefox (22%[17] do mercado), que realmente consegue concorrer com o seu equivalente proprietário (Internet Explorer - 66%), oferecendo mais segurança e adaptabilidade.

Numa observação mais geral e econômica do fenômeno do código aberto, enxerga-se a necessidade de achar modelos de financiamentos locais que permitem uma retribuição dos desenvolvedores e voluntários. O começo do projeto GNU descrevia a participação como um hobbie que podia ser retribuído com empregos ligados à área de participação voluntaria. Assim os desenvolvedores do sistema GNU/Linux usavam parte de seu tempo de trabalho como analista de sistemas, administrador de rede ou programador, para desenvolver projetos que iam dar um retorno para a empresa empregadora. Junto com o espalhamento da cultura *open-source* a outras áreas, o equilibro entre voluntarismo e retribuição não estava tão bem definido. Por exemplo, projetos como a Wikipédia que faz parte dos 10 sites web os mais visitados[18] no mundo e emprega somente 23 funcionários[19]. Nesse sentido, o movimento do software livre contribua bastante ao fenômeno de "trabalho do consumidor"(DUJARIER, 2008) fazendo o usuário-desenvolvedor criar um valor para o acionista sem custo. Num estudo sobre a nova "economia colaborativa" (*Wikinomics*), o grupo de estudo New Paradigm relevou que a peça chave desse novo xadrez é a etapa de contato entre o desenvolvedor e o projeto no qual ele vai participar. Assim plataformas oferecendo ou obrigando empresas a oferecer retribuições garantem um modelo econômico transparente enquanto projetos mais informais não podem garantir um retorno para o usuário-desenvolvedor (TAPSCOTT e WILLIAMS, 2006).

A abertura das comunidades FOSS aos investimentos privados foi feita como benefício de uma visão pragmática do movimento, ou seja, apontando suas qualidades como modelo de negócio. As empresas privadas têm enten-

[17]Fonte: http://marketshare.hitslink.com/report.aspx?qprid=0
[18]Fonte: http://www.alexa.com/topsites
[19]Fonte: http://en.wikipedia.org/wiki/Wikimedia_Foundation

dido rapidamente o interesse em favorecer a curiosidade criativa dos usuários, assim como os esforços técnicos dos desenvolvedores para adaptar os produtos às necessidades locais. Com isso, somos então afastados do conceito de *copyleft* e da filosofia do projeto GNU e os exemplos que seguem vêm ilustrar os recentes avanços da parte pragmática do movimento do software livre e os problemas que podem causar.

1.4.1 O *Software Development Kit* (SDK – Kit de desenvolvimento de Software)

O *Software Development Kit* é uma plataforma de trabalho oferecida pelo produtor de uma tecnologia para que os desenvolvedores possam criar/modificar aplicativos. Trata-se de um modelo cada vez mais usado para vários tipos de produtos: hardware, OS, jogos. Esse modelo se encontra tanto para produtos livres (Qt, Java, Android), como para produtos proprietários (Flash, iPhone, Microsoft Platform) e reutiliza as principais características do modelo de desenvolvimento colaborativo, oferecendo, ou não, o acesso ao código fonte, o programa se beneficia do esforço coletivo dos usuários-desenvolvedores.

Isso poderia ser uma redução forte do conceito de *open-source* ao benefício de uma visão muito rentável para a empresa. Deste modo, pode-se reduzir ao mínimo o compartilhamento de informação com a comunidade e aproveitar todas as vantagens de uma plataforma colaborativa. Particularmente, esse debate foi intenso no lançamento do Programa de Desenvolvimento para o iPhone de Apple (iPDP). O SDK era livre, tanto para abaixar como para usar, mas qualquer publicação de código necessitava de se cadastrar no programa oficial. Depois desse cadastramento, os desenvolvedores – mesmo sendo voluntários – tinham cláusula de segredo profissional. Embora o grande sucesso do dispositivo e de seu kit de desenvolvimento, as protestações foram tão fortes contra a política de Apple que a obrigação ao segredo foi levada, pelo menos, entre seus desenvolvedores (WILLIS, 2008).

Ademais, O *kit* de desenvolvimento de software permite às companhias facilitar a colaboração dos usuários-desenvolvedores, continuando em limitar, como elas desejam, as compensações para a comunidade. Por isso, o SDK é uma ameaça direta ao movimento do software livre e uma restrição certa ao modelo de desenvolvimento *open-source* em que seus termos, limitações e objetivos são inteiramente decididos pela sociedade emissora.

1.4.2 A computação nas nuvens

Na área de administração de redes e de gestão das informações transmitidas, dois modelos se opõem: favorecer o cliente (descentralizar a informação) ou o servidor (centralização da informação). Os detentores de informações primeiras – engenheiros, desenvolvedores, profissionais dos TI – descrevem fenômenos cíclicos na escada da administração de redes privadas (empresas, coletividades). Mas numa escala mundial, o fenômeno parece mais constante. O desenvolvimento progressivo de aplicativos Web (Web-based) força a centralização dos dados nos servidores de algumas multinacionais. O melhor exemplo, nesse caso, são os serviços *on-line* da Google. De fato, com um simples navegador, um usuário pode usar, modificar e armazenar suas fotos (Picasa, Flickr), vídeos (Youtube), blog (Wordpress, Blogger), textos, planilhas, apresentações (Zoho, Google Documents), emails (Yahoo, Gmail), itinerários (Google Maps), favoritos e rascunhos (Google Notas), contatos e agenda (Google Agenda), fluxos de informações (Google Reader), criação e hospedagem de sites (Google Page Creator), entre outros. Esse conjunto de aplicativos já excede tudo o que pode precisar um usuário de computador pessoal comum conectado à Internet. Essa situação permite afirmar que o navegador poderia se tornar a único meio necessário para usar de todos os serviços que os computadores hoje oferecem. Nesse sentido:

> Na minha opinião, eu acho que os aplicativos estão passando ao modelo Web, no qual o navegador é o principal, ou o único, software. Temos em Mozilla uma piada: 'o sistema operacional é somente um conjunto de drivers [programas fazendo funcionar os periféricos do computador, como a carta gráfica] para fazer rodar o navegador'. Dessa maneira, Windows não é melhor do que Linux ou Apple. Ele é um pano de fundo, se esquece dele. E o aplicativo para escritório, ele é nas nuvens, na Internet.[20]
>
> NITOT, 2008.

[20]"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."

Essa evolução é designada pela expressão "computação nas nuvens" (*Cloud Computing*). Ela pode tornar-se um fenômeno notável e estável da história da microinformática, como o demonstra a saída prevista do sistema Windows Cloud, SO baseado numa versão mínima do Windows permitindo de acessar os serviços online da Microsoft. Assim sendo, o seu futuro é mais provável ainda com a multiplicação dos *thin-client* (cliente leve: PDA, GSM 3ª geração, mini-notebook) cujas capacidades materiais não podem acolher os aplicativos locais tradicionais.

Embora o sucesso que teve o *kernel* Linux sobre essas plataformas portáveis, o modelo de aplicativos *web-based* ameaça as conquistas já obsoletas do "livre"em termo de aplicativos locais (Gimp, Open Office) oferecendo ambientes intuitivos e acessíveis de qualquer lugar conectado, mas cujo código é fechado.

1.4.3 Google

Os atores privados tiveram um papel determinante no desenvolvimento dos softwares de código aberto e é graças a esses investimentos que muitos deles ganharam em credibilidade. Empresas como IBM (Linux), SUN (Open Office, Virtual Box, BSD), Red Hat (Fedora), Canonical (Ubuntu) participam desse esforço em função dos interesses e de estratégias próprias, mas nenhuma contribui ao fenômeno, de maneira global, como o faz o Google desde sua criação.

Criada em 1996 como simples motor de pesquisa, o algoritmo de busca deu resultados inesperados o que permitiu a empresa crescer rapidamente e de se tornar, em apenas 10 anos, um dos atores principais das novas tecnologias. Graças a um sistema de propaganda discreto e dedicado pelo exame das informações deixadas por cada usuário, a empresa tem agora um capital financeiro e uma influência consideráveis. Segundo a *Harvard Business Review* (IVEN e DAVENPORT, 2008), são milhares de experimentos diários efetuados em cima do maior banco de dados sobre como as pessoas procuram, acham e usam as informações. Além disso, eles detêm o maior número de perfis e de informações sobre os usuários da internet no mundo, o Google mesmo se apresenta como: "no Google, nossa missão é organizar toda a informação do mundo"[21].

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

Tal influência e essa capacidade de investimento - perto 30 bilhões de renda - têm contribuindo bastante ao desenvolvimento e a criação de numerosos projetos *open-source*. A própria Google libera funcionários para adiantar certos projetos e financia diretamente alguns aplicativos chaves, além de organizar uma grande rede de colaboração, na qual estudantes do mundo inteiro podem prestar candidatura para participar de projetos abertos apoiada pela empresa (*Google Summer of code*). Também, alguns de seus projetos são disponibilizados em código aberto (Android, Chrome).

Para terminar, as opiniões sobre as contribuições de Google à comunidade SL/CA são radicalmente divididas e ilustram bem as oposições entre adeptos do "*free*"e adeptos do "*open-source*". Para os mais radicais e comprometidos a preservar as raízes éticas e ideológicas do movimento, "Google is evil"[22] (Google é o diabo) e encerra o movimento nos meios-termos já realizados pela asa pragmática, mas sob controle de um monopólio de empresa. Para os pragmáticos do movimento, o apoio é precioso e as transgressões à GPL já são freqüentes. Porém, todos ficam preocupados em observar certas contradições, indo no sentido duma estratégia de controle e de monopólio. Do ponto de vista da empresa, parece que a participação inédita que eles dão às comunidades SL/CA é tanto chave de seu sucesso, quanto um risco em ver essa mão de obra, sobre a qual não tem controle, sumir ou mudar os termos da sua produção (FABERNOVEL, 2008 e 2009).

[22]Assunto duma mailing-list do grupo Digital Freedom global Activists (DFGA – binaryfreedom.info)

Capítulo 2

ASPECTOS SOCIAIS DO MOVIMENTO DO SOFTWARE LIVRE

Neste capítulo tem como propósito apresentar os elementos políticos e sociais do movimento SL/CA presente na literatura universitária desenvolvida a seu respeito, e propor uma tipologia deles. Depois de apresentar os termos usados para designar os atuantes do movimento do software livre dentro das comunidades e nos sentidos comuns e universitários (2.1.1) abordaremos o movimento através do conceito de ética hacker (2.1.3), entendido num contexto teórico de construção social da ética (2.1.2). Uma crítica construtiva desse conceito permite reinterpretar as significações culturais do hacking (2.2) pela noção de "agnosticismo político"(2.2.1). Isola-se então a paixão para a informação como matriz mínima das interações de natureza política dos usuários-desenvolvedores, permitindo então construir uma tipologia dela a partir dos trabalhos da antropóloga Gabriella Coleman (2.2.2).

A partir disso, construiremos um paradigma de interpretação da atividade do usuário-desenvolvedor de SL/CA através da idéia de pragmáticas da programação (2.3). Após ter apresentado o ato de programar na sua dimensão artística e reguladora (2.3.1) poderemos usar das pragmáticas de programação como a interpretação sociopolítica de um conceito lingüístico, permitindo interpretar as relações dos usuários-desenvolvedores com suas políticas informais e a informação (2.3.2).

2.1 A Ética Hacker

2.1.1 Hackers, usuários-desenvolvedores, geeks e nerds

Numerosos termos designam as pessoas nos arredores do software livre. Porém, significações suplementares vêm apontar categorias mais específicas ou gerais de indivíduos. Por essa razão, apresentaremos, com detalhes, quatro designações que foram encontrados com muita freqüência ao longo das pesquisas bibliográficas e de campo.

Hacker

"Hacker"é provavelmente o termo o mais romanceado a respeito da cultura informática. A indústria cinematográfica de Hollywood já se aproveitou dos recursos cognitivos que se encontram por volta deste domínio: um "ego ilimitado"exprimido em um mundo de 0 e 1 enfrenta uma realidade manipulada, truncada, decepcionante, limitada. A trilogia dos irmãos Wachowski[1], *Cypher*[2], *Pi*[3], desenvolvem essa mesma idéia, sem contar com as inumeráveis produções de segundo plano, cuja lista não pode ser exaustiva[4].

Este termo, hacker, é quase sempre apresentado como derivado do termo inglês *hijacker*, pirata. Com menos frequência, sublinha-se a sua relação etimológica com o verbo cortar nas línguas germânicas, e também em inglês. Enquanto a etimologia inglesa faz referência ao sentido muito midiático de bandido explorando sistemas informacionais e redes de computadores (*Cracker*), a etimologia germânica nos abre a um horizonte mais amplo de significados. De fato, trata-se então de "cortar"os caminhos já conhecidos para achar dicas, soluções, hipóteses para um problema que sejam originais, não óbvios e elegantes. É nesse sentido que o termo *hack* foi usado pela primeira vez, dentro dos laboratórios do clube do *Tech Model RailRoad* (TMRC) do MIT na década de 50. Assim, chamava-se de "*hack*"as modificações inteligentes que faziam nos relês eletrônicos. Também, na década de 60, fazia-se referência ao ato de hackear (*hacking*) dentro do laboratório de inteligência artificial

[1] *The Matrix* (1999), *The Matrix Reloaded* (2003), *The Matrix Revolutions* (2003), por 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 outros.

da mesma universidade, onde foi fundado o projeto GNU. Apontava então o fato de ir procurar o código fonte para melhorar e adaptar um software para seus próprios fins. Nesse sentido, hackear relaciona-se particularmente com o movimento do software livre. As empresas brasileiras de informática como Google Brasil, Microsoft Brasil, IBM Brasil, sempre fizeram referência ao *Decifrador*, em respeito à língua portuguesa.

Geek e Nerd

Geek é uma "gíria"inglês derivada da palavra *geck*, idiota (*fool*) e estranho/marginal (*freak*). Também o termo geek tem origens com a palavra *geck* do germânico antigo, significando louco, e em dialetos franceses, *Gicque*, significando "louco de carnaval"[5]. Trata-se de "uma pessoa esquisita, ou pelo menos estranha, particularmente uma sendo percebida por ser demais obsessiva com uma ou várias coisas dentro das quais a intelectualidade, a eletrônica"[6]. Podemos qualificar a cultura *geek* como movimento cultural popular típico da sociedade de informação contemporânea. Além disso, os geeks são qualificados como tendo interesses exaustivos e criativos sobre assuntos relacionados como novas tecnologias (KONZACH, 2006). Por isso o software livre pode ser considerado como uma área da cultura *geek*, e as pessoas participando por volta dele como *geeks*. Assim, o ato de programar encaixa-se perfeitamente nas exigências de exaustividade, de criatividade e de mudança presentes na cultura *Geek*.

O antropólogo Christopher Kelty, no seu estudo sobre as significações culturais do *free software*, faz constantemente referência aos *geeks* como sendo os atuantes principais do movimento (KELTY, 2008), e para ele, trata-se de uma convergência filosófica pelo lado do transumanismo. Este transumanismo é uma modalidade de crença na linha do tempo do progresso técnico, a qual permitiria ir além do corpo humano. Para os transumanistas, a intervenção técnica tem um papel específico que a diferencia da intervenção política, legal, cultural ou social. Nesse âmbito, não tem retórica, mas uma hipotética "verdade pura" do "funciona!". No extremo e para usar de uma metáfora própria ao mundo da programação: "as idéias ruins não se compilarão[7]".

Embora tivesse uma conotação de "bizarro" por volta do termo *geek*, pa-

[5]Fonte : Oxford American Dictionary

[6]Dictionary.com

[7]Um programa é "compilado" quando uma serie de comandos e instruções se tornam um aplicativo executável.

rece que a conotação pejorativa fica mais na palavra *Nerd*. De fato *Nerd* designa uma pessoa de alta capacidade intelectual e tendo interesses fora dos padrões sociais comuns. Assim apontam-se mais os significados de desvio, estranho, fora do padrão. No sentido comum, parece que a figura do *geek* constrói-se no âmbito de um especialista autodidata, enquanto a do *nerd* desenha a de um adolescente sarapintado. Na verdade, essas diferenciações aparecem quando se procura definir categorias, pois a guerra *Geek-Nerd* não tem propósito por seus atores, e cada um pode chamar o outro sem ter uma identidade muito diferente. O termo pode ainda ser usado da mesma forma que *geek* para designar atores do software livre, como o *Nerd* é o ideal-tipo do hacker usado por Paul Graham em *Hackers and Painters* (GRAHAM, 2004), por exemplo.

Usuários-desenvolvedores

Por fim o termo de usuário-desenvolvedor é o único dos termos revistos aqui que é específico às comunidades de software livre. Designa as pessoas que por volta dessas comunidades usam softwares livres, e por esse uso chegam a ajudar ao desenvolvimento do próprio software. Essa ajuda pode ser mínima, ou quase simbólica, aceitando mandar "*crash report*"[8] ou de responder a questionários. Ademais, pode tratar-se de tarefas muito maiores como o estabelecimento de documentação, programação de módulos adicionais, desenvolvimento do próprio software.

A figura do usuário-desenvolvedor é muito presente nos projetos livres, eles constituem a "comunidade" que define esse modelo de desenvolvimento como colaborativo. E é provavelmente na reprodução dessa figura em outros domínios de produções que se percebe a influência que pode ter o movimento do software livre fora de suas fronteiras. Isto é, os padrões de "consumidor-produtor" da economia colaborativa (Wikinomia) e de "leitor-redator" da produção de conteúdo na Web 2.0 que são intimamente ligadas ao imaginário do software livre e a figura do usuário-desenvolvedor (TAPSCOTT e WILLIAMS, 2006).

2.1.2 A construção social da ética

Uma das análises bastante desenvolvidas a respeito dos aspectos culturais do movimento do software livre e da cultura hacker é a da ética. Assim, a

[8] Relatórios de mau-funcionamento

"ética hacker" é um paradigma suficientemente flexível para integrar a multidão de casos que formam as comunidades por volta de cada software. Uma demonstração disso é que se hacker e usuário-desenvolvedor não se alinham exatamente nas mesmas realidades, a cultura hacker é um dos pontos comuns a todas as comunidades de usuários-desenvolvedores.

Para trazer ao conceito de ética a flexibilidade desejada, deve-se afastar a concepção kantiana da ética, segundo a qual a lei moral vem de nenhuma forma da experiência empírica, mas sim de imperativos categóricos, pré-estabelecidos: "Age de tal modo que a máxima da tua ação se possa tornar princípio de uma legislação universal". Trata-se para nós de tomar conta da "vida social da ética" no sentido dos teóricos sociais como Bakhtin (BAKHTIN, 1984, 1993). Assim, as regras de vida e de convivência constroem-se interativamente e dinamicamente a partir das práticas sociais e das experiências vividas no quadro comunitário. Desta maneira, a ética pode se manter somente observada a uma relativa estabilidade das práticas sociais e uma comunidade de sentido por volta das experiências vividas. A análise se foca, então, sobre eventos que geram transformações, e desse modo, aponta-se a importância da auto-modelação comunitária, ou seja, do papel decisivo do contexto social, histórico e econômico.

Nessa perspectiva, a idéia é simples e desenvolvida por numerosos teóricos (GALISON, 1997; GOOD, 1994; GUTERSON, 1996), pois se trata de excluir os excessos de particularismos como os de universalismos e ficar focado na observação dos acontecimentos concretos. Assim, os indivíduos adotam valores e realizam escolhas morais através de ações que são influenciadas por instituições ou tecnologias. É isso que podemos aprender das palavras de Coleman e Hill: "As experiências sociais que vêm a favor ou contra uma prática ética, modela a natureza das relações individuais e determina as orientações éticas de uma comunidade" (COLEMAN e HILL, 2005). Em outras palavras, o uso e a organização dessa ética interagem com o meio da comunidade ou do grupo em escadas econômicas, políticas e sociais. Em nível econômico, pode se tratar da produção de um bem, de um modelo de desenvolvimento ou de organização. Já no político, pode ser sua agenda política e suas manifestações públicas. Por fim, no âmbito social, uma rede e nós de indivíduos e grupos estendendo-se e traduzindo suas normas em contato com novos domínios e grupos. Por nos permitirmos fazer o laço entre o modelo de desenvolvimento participativo, alguns crimes eletrônicos, as conferências internacionais sobre o Open-Source, Internet, etc, a 'ética hacker' é uma noção heurística ao entendimento do fenômeno do software livre.

2.1.3 O conceito de ética hacker

Contam em literaturas que a expressão "ética hacker"foi usada pela primeira vez pelo jornalista Steven Levy, em uma das primeiras obras reconhecida sobre as cybers-comunidades: *Hackers, Heroes of the Computer Revolution* (LEVY, 1984). Em contraste com as definições usadas logo após, Levy não hesita em dar ênfase a seu conteúdo. Segundo ele, a ética hacker é um tipo de predecessor moral ao software livre e às comunidades open-source. De acordo com essa interpretação da ética do *hack*, o acesso a informação é total. Qualquer acesso a um conhecimento faz descobrir uma parte do mundo, e por isso deveria ser absoluto, sem limites. Por isso, a informação e sua circulação devem ser livres. Para livrar-se dos constrangimentos institucionais, precisaria desafiar as autoridades estabelecidas e ficar desconfiado na frente de legitimidades exteriores às comunidades. Nesse sentido precisa privilegiar formas descentralizadas de organização e, nesse ambiente, os hackers devem ser apreciados em função de suas qualidades técnicas e criativas e não por qualquer critério social (raça, idade, sexo). Por fim, tem-se a idéia que o fortalecimento técnico e prático das tecnologias informáticas contribuirá para um "mundo melhor", uma melhor organização da sociedade, mais repartida, justa, democrática.

À propósito dos "verdadeiros hackers"que Steven Levy queria identificar com essa definição, achamos: John McCarthy, Bill Cosper, Richard Greenblatt, Richard Stallman. Depois o autor descreve uma "segunda geração'"de hackers, os *hardware hackers*. Tratam-se dos grandes nomes da computação pessoal cuja maioria está envolvido no início do Vale do Silicone: Steve Wozniack, Steve Jobs, Bill Gates, Bob Marsh, Fred Moore. Por fim, uma terceira geração aparece na época dos primeiros jogos interativos, os "games hackers"(John Harris, Ken Williams, entre outros).

Todavia, concentrando-se sobre as produções ganhadoras das comunidades de informáticas (Software, Hardware, Jogos), o autor de *Hackers* parece passar ao lado do fenômeno social dos agrupamentos de hackers como um todo. De fato, mesmo em 1984 - data da publicação do livro - as práticas hackers ultrapassavam já muito os quadros industriais e contribuíam à construção e à publicação de uma quantidade imensa de informações que ia permitir a qualquer um, a seu nível, de participar/aproveitar desse progresso técnico, da era da informação e das redes mundiais. No caso do software livre, atribuir a Richard Stallman ou a Linus Torvalds a paternidade do sistema GNU/Linux não é um erro somente político de esquecimento das massas de

engenheiros que contribuíram ao projeto, mas também afasta o observador ou o leitor dos interesses, laços e valores que motivou milhares de hackers do mundo *online* a trabalhar juntos. Isso representa de aceitar como material de análise e de testemunha dessa época os *mailing-lists*, fóruns, *wikis*, manifestos, debates, fluxos de notícias que fizeram a cultura hacker e estruturaram sua ética.

Desse modo, o filósofo finlandês Pekka Himanem desenvolveu uma noção de ética hacker diferente. Ao fazer referência aos trabalhos de Weber sobre a ética protestante e o capitalismo moderno, ele opõe os valores hackers aos valores protestantes. Segundo ele, a ética hacker aproxima-se mais das definições da "atividade"como se pode achá-la a obra de Platão ou Aristóteles: um trabalho de excelência por volta de uma procura de paixão, de felicidade e de criatividade (HIMANEM, 2001).

As análises de Himanem como as do Levy, todavia, parecem se cristalizar por volta de um contrapeso aos desenvolvimentos pejorativos da imagem do pirata informático, e nesse sentido, achamos trabalhos ainda mais explícitos tentando "resgatar"uma fama ridiculizada (BEST, 2003; HANEMYR, 1999). De modo geral, as concepções do hacker e da hacking as quais esses trabalhos tentam se opor são, por um lado, adolescentes obcecados pela internet e pela conquista de saberes perigosos e/ou proibidos (BORSOOK, 2001; SLATALLA e QUINTER, 1996), e por outro, torneios audaciosos de intrusão em sistemas privados (SCHWARTAU, 2000). De fato, parece que os estudos até então citados se entregam a interagir com os preconceitos existentes a respeito da figura do hacker e de seu papel na sociedade da informação, sem nunca se perguntar o que faz a particularidade em si desse grupo. Nesse sentido, a antropóloga Gabriella Coleman, cujos trabalhos vão fazer referência ao longo do nosso estudo, chegou a afirmar: “A literatura sobre Hackers, pois, tem tendência a encerrar o ato de hackear num binário moral no qual hackers são ou louvados, ou desprezados. Esse tendência ameaça isolar mas do que esclarecer a significação cultural do computer hacking.”[9] (COLEMAN, 2008, p.256).

[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.”

2.2 As significações culturais do hacking

> "Voltada a sua expressão a mais simples – e abstração feita de seu conteúdo – a ética hacker tem seu equivalente na formula l'art pour l'art (a arte pela arte). O importante é de entender que ao contrario do ativismo político, o objeto da atividade do hacker, o conhecimento e o exercício da curiosidade, é interior ao sujeito."[10]
>
> REIMENS, 2002

2.2.1 O agnosticismo político do Movimento do Software Livre e da figura do Hacker

É indubitável que numerosos projetos foram inspirados pela filosofia do "livre". Na área do jornalismo, disponibilizaram-se arquivos inteiros (ex: BBC) e desenvolveu-se a figura de um leitor participando – leitor-redator – à construção e discussão dos fluxos informativos, especificamente nas tendências da web 2.0. Na área da lei, acham-se formas contratuais novas para regimentar as produções intelectuais e artísticas inspiradas pelas licenças livres (ex: *Creative Commmons*). Na educação, cursos são disponibilizados *online* como é o caso no MIT com o OpenCourseWare, e isso por cima de tecnologias livres. Esses projetos são todos ligados a um contexto de rede e de interconexão oferecido pela internet e a sociedade de informação. Mas como o afirma Christopher Kelty, as significações culturais do software livre vão além do simples diagnóstico da "sociedade da informação", pois provém de uma reorientação mais específica, portanto tem a ver com práticas técnicas e legais detalhadas e precisas. Também se trata de uma reorientação mais geral porque cultural e não somente econômica ou legal. Uma prova disso é que com a Internet, a governança e o controle da criação e da disseminação do saber mudou consideravelmente, operando assim uma "reorientação do poder e do conhecimento" que questionam profundamente a relevância e a legitimidade do sistema de propriedade intelectual (KELTY, 2008).

[10] "Ramenée à sa plus simple expression, - et abstraction faite de son contenu – l'éthique hacker a son equivalent 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"

Nesse contexto, embora seja claro o fato de que a vida política do SL/CA, por suas atuações e influencias, já é avançada, a figura do hacker e dos atuantes do movimento continua recusar qualquer afiliação política. É isso que nota Gabriella Coleman ao dizer: "Enquanto é perfeitamente aceitável e incentivado de ter uma mesa sobre software livre numa conferência antiglobalização, desenvolvedores de SL/CA recomendam que seja inaceitável de reivindicar que o SL/CA tem como propósito a antiglobalização, ou a respeito disso, qualquer programa político."[11](COLEMAN, 2004, p.507).

Cabe aqui pensar que esse "agnosticismo político" pode exprimir-se através de varias defesas contra as tentativas de dar um sentido político a um ato, uma produção ou uma situação. Na área de segurança informática, o critério principal é a perfeição do sistema de proteção, e por isso as considerações de ordem ideológicas são afastadas com a observação de que a tecnologia a ser usada pode ser livre ou não, enquanto for aberta a atualizações rápidas e com desempenho estável. Cabe aqui acrescentar o que um participante do canal *#hack* do servidor IRC Freenode comentou durante um debate sobre o *free software*: "não é bem 'tanto faz', assim... se as ferramentas forem livres, melhor, que a gente trabalhe com sistemas abertos [freeBSD, Linux], mas o importante é ter a tecnologia mais segura, mais configurável, mais controlável. É isso que faz de uma tecnologia que ela seja boa para um trabalho de segurança, e não o fato dela ser livre ou não."[12]. Assim o site *insecure.org* que mantém uma famosa lista de 10 melhores ferramentas de segurança informática[13], sequer comenta o tipo de licença que acompanha os softwares. Ademais, os critérios privilegiados são o preço (gratuito ou não), a compatibilidade com diversos sistemas (Linux, Windows, OSX e BSDs) e o acesso ao código fonte. O software que chega em primeiro dessa lista desde anos, Nessus, uma ferramenta para identificar falhas de segurança, é de código fechado e pago.

Em atuações mais diversificadas da computação, incluindo administração de redes, análise de sistemas de informações, programação web, o recuso da ideologia manifesta-se muito ao encontro da figura do responsável de projeto, cujas capacidades técnicas são frequentemente reduzidas, e aos efeitos

[11]"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."

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

[13]http://sectools.org/

de moda pelos quais eles passam. Ao longo do nosso trabalho, encontramos numerosos testemunhos nesse sentido, particularmente na área de desenvolvimento web, onde os programadores convivem com superiores voltados ao marketing e à comunicação publicitária. A ideologia aparece, então, como um meio para o ignorante da técnica de se familiarizar com as limitações e do executante da tarefa. Assim, ele constrói um discurso permitindo justificar as opções tecnológicas, de orçamentos, de prazos, entre outros. Chega-se, desta forma, numa situação onde o responsável de projetos, muitas vezes numa situação hierárquica ascendente, vai realizar escolhas a partir de informações truncadas que os técnicos vão seguir sem poderem se preocupar em saber quais são as melhores opções a seguir para garantir o melhor serviço. Podemos identificar isso na testemunha de Marcelo, administrador de rede em uma universidade publica brasileira, ilustra:

> Devíamos programar o troço para a SOFTEX[14]... 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%...

Ao contar esse evento, o informante dá um exemplo de situação aonde a ideologia ("a propaganda TI") vem prejudicar o trabalho dele e obrigá-lo a realizar uma tarefa com opções tecnológicas que não são as melhores, isso por razões que parecem "absurdas", ou seja, "crenças" de alguém com posição hierárquica superior e com um conhecimento técnico incompleto e influenciado por efeitos de moda. Aqui "o cara da SOFTEX" deve provavelmente basear sua escolha da linguagem de programação JAVA sobre tendências internas a sua instituição de promoção do Software Brasileiro ("o JAVA é

[14] Associação para Promoção da Excelência do Software Brasileiro - SOFTEX

entendido como o 'melhor', então um software feito com JAVA 'é' melhor") desligando-se do raciocínio lógico voltado ao desempenho da tecnologia.

Embora esse exemplo ilustre uma polêmica técnica que não pode ser chamada diretamente de "política", parece que é a mesma lógica que determina a consciência política desses atuantes. Assim é que a figura do responsável permanece nas lógicas da sua instituição favorecendo seus interesses próprios ou pelo menos interesses que não são os dos profissionais. Outra questão colocada é que a idéia de exigência de excelência técnica cria um raciocínio que protege os atores do político, nesse sentido, o movimento SL/CA não pretende ganhar uma decisão política no seu favor ou conquistar políticas públicas, mas continuar evoluindo com uma autonomia própria permitindo o melhor desenvolvimento técnico.

Em nosso entendimento, essas observações permitem conhecer melhor o fato que ambas as literaturas, anarquistas e liberais, fazem referência ao movimento do software livre como exemplar do seu modo de produção e de organização. Por um lado, as análises liberais acham aqui um exemplo concreto de "Mão invisível", de ausência de intervenção do estado ou de qualquer outra autoridade externa nos processos de decisão internos. Os atores ainda desenvolvem uma consciência própria do ambiente deles por um processo livre e conseguem tomar decisões, para fazer escolhas voltadas a seus interesses próprios que, juntos, criam e mantém a comunidade do software livre. Somado a isso, temos pensadores que acham realizações concretas de formas anarquistas de organizações no movimento SL/CA e suas comunidades (MOGLEN, 1999, GROSS, 2007). Sublinham-se, então, a alta produtividade real da autogestão de trabalhadores, e as raízes lógicas do desafio às autoridades. Assim, mantendo uma possível alternativa ao neoliberalismo e ao anarquismo, como uma reformulação deles, o agnosticismo político do movimento SL/CA e da figura do hacker protege sua exigência de excelência técnica de direções políticas predeterminadas.

2.2.2 A informação como paixão e a tipologia das políticas informais

Tentando achar a essência do hacking, ou seja, um ponto fixo de análise razoavelmente independente dos preconceitos dos sentidos comuns, a antropóloga Gabriella Coleman vai focar-se sobre a relação que mantém os hackers com a informação. Assim, uma semelhança de todos os usuários-desenvolvedores,

hackers e *nerds* é o amor dado à liberdade de informação e sua livre circulação (COLEMAN, 2003). Trata-se de um sentimento muito forte, maníaco e estético próximo ao *amour fou* (amor louco) descrito nas obras do anarquismo ontológico, como o meio de realizar-se inteiramente como indivíduo e desafiar as estruturas das sociedades (BEY, 1985 e 1991). Segundo as palavras de Bruce Sterling, os hackers são "possuídos não meramente por curiosidade, mas por uma verdadeira luxúria do saber."[15]. Tentando achar as origens de tal paixão, Gabriella Coleman compartilha sua experiência:

> Minha experiência com o software livre defende esse principio. O espírito de exploração que forma as bases do hacking deve começar em desmontar a mistura familiar, pelo horror da mãe: depois traz a aprender a programar com cinco anos, pela alegria da unidade parental, depois se transforme em se trancar no quarto para ler todos os manuais de computação, os quais os pais confundem com inquietude pré-adolescente; depois é aprender cada entrada e saída desse sistema operacional simples chamado de Unix, descobrindo cada um dos traços tipográficos e temporais da Internet, pelo espanto do antropólogo; e finalmente passar seu tempo contribuindo, escrevendo código para projetos de códigos abertos, muitas vezes, de novo, pela consternação de seus pais.[16](COLEMAN, 2003, p.298)

Depois de toda esta discussão, vale fazer referência a nossa experiência dentro de um departamento de administração de redes e sistemas de uma universidade pública brasileira, como o contato com numerosos profissionais e usuários desenvolvedores de vários domínios da informática. Isso nos permite afirmar que o elemento de paixão pela informação é determinante na diferenciação do "exército de reserva"dos profissionais das TI, interessados pela aquisição de um *know-how* e de seu valor direto no mercado do trabalho, e os que, em cima dessas preocupações materiais, desenvolvem um

[15]Citado em COLEMAN, 2003.

[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 transforminto 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."

interesse constante para a apropriação, a produção e a discussão da informação. Dentro desse departamento de informática, a divisão das tarefas parecia ser mais determinada pela capacidade de cada um "brincar" na frente de um problema que por formação ou cargo oficial. O termo de "brincar", usado pelos próprios atuantes, refere-se principalmente às aptidões dos funcionários a responsabilizarem-se, ou seja, a enxergar os ramos do problema enfrentado com outros possíveis erros e tentar resolvê-los do jeito o mais sistemático, e de surpreender-se, ou seja, jogar com os paradigmas utilizados naquele processo e cruzá-los com considerações técnicas e teóricas para melhorar sua concepção geral do sistema. Desta forma, a curiosidade e o interesse pessoal para o funcionamento das tecnologias utilizadas parecem ser o critério determinante para julgar da qualidade de um funcionário. Um informante ilustra:

> Cada vez que um novo estagiário entra [no departamento de informática] eu falo o seguinte: a Universidade não vai te aprender um trabalho mas te dar uma formação geral... Você vai aprender as coisas colocando as mãos na massa, com problemas técnicos, concretos, procurando soluções, jeitinhos simples e eficazes. Mais você fica procurando as razões atrás do problema, ou seja, as estruturas do sistema, como funciona uma rede, um protocolo, como dialogam dois computadores, mais você vai criando concertos expertos, inteligentes, que vão ficar, em cima dos quais a gente vai trabalhar e desenvolver novas coisas. Isso, poucos entendem, nhé? A última chamada para estagiário teve mais de 15 no meu escritório e nenhum deles soube me dizer o que é um banco de dado... Acredita?

Ao entender essa característica do hacker, seja técnico de laboratório ou usuário-desenvolvedor num projeto livre, permite enxergar o que define mais essa figura é sua procura "absurda" para uma informação livre, abundante e de precisão, e é a partir disso que se constrói a interação do atuante com seu contexto social. O usuário-desenvolvedor quer uma informação livre capaz de circular e de ser modificada, mas isso acontece em um ambiente de micro-restrições que limitam sua atividade. Portanto, falamos no trato de restrições técnicas, legais, culturais, políticas, econômicas, que perturbam uma atividade subjetiva de auto-realização. A respeito do software livre, as restrições mais óbvias são as legais, ou seja, dentro do código, aquele que se pode abrir e modificar, e aquele que somente pode executar. Dentre outras

opções, a engenharia reversa permite obter parte de um código fechado, porém ultrapassam-se os limites dos direitos autorais, como o caso de usos de inovações livres para melhorar tecnologias fechadas, por exemplo, isso ilustra o fato que a fronteira entre software livre e pirataria não é tão nítida que ambos dos atores – livre e proprietário – o deixam entender. Além de ter numerosos protocolos e ferramentas de pirataria com licença livres (Bittorrent, *Peer-to-peer*) a própria raiz do movimento SL/CA foi de certa forma o *hacking* do sistema UNIX da AT&T (cf. 1.2.2). Essa confusão pode até ser explorada pelas próprias companhias de software proprietários, e com essa preocupação, Túlio Vianna observou que os relatórios americanos sobre a pirataria mundial, principalmente da *United States Trade Representative* (USTR), juntam os *marketshares* do software livre com os da pirataria para denunciar os crimes contra os direitos autorais (VIANNA, 2006). Mas o contexto de restrições que encontra o *geek* na sua atividade de busca vai bem além das considerações legais que ficam as mais mediáticas, porque há a criação de vários genéricos concretos de interação com a sociedade e suas estruturas. Nesse sentido, Coleman identificou vários tipos ideais de lógicas especificas às éticas da informação diferenciadas, é o que pontuaremos a seguir:

As políticas de transgressões: o *underground hacking*

A parte transgressora do mundo do *hacking* é seu rosto fantasiado e mediático, e também desprezado. Em todos os casos, ela fica quantitativamente, em termos de produção de código, de participantes e de instituição, minoritária. O *hacking underground* ilustrou-se particularmente na formação dos conceitos de engenharia social e de *human data* (dados humanos) onde a sistematização do comportamento humano e sua piratagem expressam-se através de um cinismo explícito (MITNICK e SIMON, 2002). Isso significa uma crítica forte do liberalismo associada à noção Nietzschiana do poder e do prazer. Mas "como a tentativa de Nietzsche de elevar o poder criativo do individuo nunca conseguiu verdadeiramente escapar-se das noções liberais do iluminismo, a prática do *hacking underground* representa mais uma radicalização dos fundamentos do liberalismo do que um verdadeiro cisma" (COLEMAN e GOLUB, 2008, p.263). Isso acontece devido a uma ética da transgressão como crítica política que se manifesta pelo culto "prazer de ser vigiado" e da "interface entre o vigiamento e escapamento ao vigiamento" (HEBDIGE, 1997). A informação é "boa", "agradável" se ela é proibida e que

sua aquisição necessita transgressão.

As políticas de tecnologias: a criptoliberdade

A criptografia é a cifragem de dados por um algoritmo reversível por meio de uma chave de dados secundária (decifragem). Essa tecnologia é usada com fins de confidencialidade, de autenticação e de controle de integralidade. Na área da informática, a primeira chave pública foi publicada em 1975 por Whitefiels Diffie e Martin Hellman (MIT), que revolucionaram então o domínio da criptografia. O uso da criptografia estende-se, então, às instituições, às empresas, por exemplo, mas não existem ainda soluções para os computadores pessoais.

Em 1991, enquanto o senado americano ia votar uma lei para proibir o uso privado da criptografia, Phil Zimmerman publica a primeira chave publica utilizável num computador pessoal (PGP – *Pretty Good Privacy*), tornando-se então culpado de um ato de desobediência civil e arriscando acusações por traição. Ainda, o autor do programa desenvolve assim todo um discurso durante sua defesa na cena mediática, vejamos:

> Se a privacidade é fora-da-lei, somente os criminosos terão direito a privacidade: [...] PGP permite às pessoas de encarregar-se da sua privacidade. Há uma carência social crescente para isso, é por isso que eu o escrevi.
>
> Phil Zimmerman

Como observamos anteriormente, isso é o início da ética criptolibertária, que ficou depois fortalecida pela criação em 1992, dos Cypherpunks, associação de hackers militantes para os direitos civis. Eles trabalham em cima das tecnologias de privacidade e militam contra as leis limitando a privacidade individual, o que quer dizer que valores liberais, no plano político, sustentam esse movimento, notavelmente as de autonomia do indivíduo e de sua liberdade a respeito do governo, pois se cria a expectativa que as tecnologias podem resolver problemas sociais, porque reformulam na nova linguagem tecnológica a repulsão liberal a intervenção do estado. Concretamente, esse movimento junta anarcho-capitalista radicais, democratas, republicanos, e, para alguns deles, nada há de novo nesse movimento, isto é, somente a continuação de uma luta para o direito constitucional.

Ademais, a área da criptografia é um bom exemplo de influência das comunidades livres nas estruturas da sociedade, tornando difícil a interdição do uso da tecnologia disponibilizando-a. Assim, alguns países como a França, onde a criptografia foi proibida de principio, chegaram a autorizá-la (janeiro 1999) por necessidades provocadas pelas realizações concretas das comunidades (uso livre, necessidade social).

As políticas de inversão: o movimento do software livre

Paralelamente ao movimento criptolibertário, outros hackers desenvolvem outra visão da segurança. Para Richard Stallman, fundador da FSF, o conhecimento de maneira geral, não deve ser objeto de qualquer orientação porque o benefício tirado de uma informação é sempre feito no detrimento da comunidade. Stallman militava dentro dos escritórios do MIT, deixando sua máquina sem nenhuma senha para os arquivos a serem disponibilizados para todos com uma mensagem de boas vindas explicando sua filosofia da informação. Quando a FSF foi criada em 1984, desenvolveu-se uma pedagogia que abriu o mundo do hacker para fora. Suas ações respeitavam a lei e se servia dela para proteger-se. A criação da licença pública (GPL) delimitou uma zona legal de segurança, de publicidade onde códigos ficariam abertos. E um conceito de liberdade positiva que é colocado na frente, de liberdade pela abertura, e não uma liberdade negativa, pelo fechamento ou o segredo como é o caso com a criptografia.

Essa idéia de inversão acha-se também no artigo *Beating Them at their Own Game* (ganhar ao jogo deles) (BEST, 2003) onde Kirsty Best demonstra que o SL/CA representa para seus usuários mais um investimento do que um rejeito das estruturas sociais existentes nos arredores das novas tecnologias. Não se trata de promover uma mudança radical, mas de "entrar no jogo" das estruturas capitalistas para redefinir seus termos a respeito das tecnologias da informação e dos novos meios de comunicação.

As políticas de colaboração: o movimento *open-source*

Esse tipo ideal foi adicionado em nosso trabalho à tipologia da Gabriella Coleman, com o objetivo de marcar a diferença entre a ética *open-source* e a "livre", e para sublinhar a convergência de várias éticas dentro dessa.

Desse modo, associado por "coincidência" (TORVALDS e DIAMONDS, 2001) ao projeto GNU, Linux e seu projeto tecnológico vão favorecer a va-

riação "open-source" do movimento do "livre" chamado a se radicalizar. O Software de código aberto, promovido antes de tudo como modelo de desenvolvimento não é mais somente "bom", mas principalmente eficiente. A liberdade de informação libera um *entertainment*, um mérito que são motivações muito mais eficazes que um simples salário para incitar à participação a um espaço colaborativo produtivo. Essa ética hacker é muito desenvolvida na obra de Himanem, prefaciada pelo criador de Linux, onde o foco é feito sobre a flexibilização do trabalho, e considerado como *hobby* e paixão antes de tudo, sem remuneração direta sistemática (HIMANEM, 2001). A ideologia transmitida aqui é mais econômica do que política, e, além disso, a liberdade de informação é necessária porque ela é estratégica. Para tanto, precisa-se favorecer a criação de valores e a liberdade de fazer um benefício dela. Nessa nova economia colaborativa, "inteligente", a "wikinomia"[17], "a capacidade de juntar os talentos de indivíduos e empresas dispersadas está se tornando a competência chave do dirigente e da empresa" (TAPSCOTT e WILLIAMS, 2007) e, por assim afirmado, quase o objeto do próprio hack. Na área do software, esse grupo ilustrou-se na construção da web 2.0 pelas ferramentas participativos onde os usuários doam conteúdo a plataforma, como é o caso da enciclopédia Wikipédia baseado na tecnologia Wiki, software livre desenvolvido por Ward Cunningham em 1995. Ainda, a ética com dominante colaborativa ilustra-se na esfera tecnológica com a codificação de algoritmos performáticos, visando criar estatísticas dedicadas à análise dos comportamentos dos usuários da internet. Isto é "programar a inteligência coletiva"[18] e tratar as informações acumuladas pela observação dos usuários, para, por exemplo, sugerir "produtos assimilados", "par perfeito", etc. Se a programação da colaboração desenvolve sua própria ética da informação, suas ferramentas tecnológicos inspiram-se muito dos movimentos vistos precedentemente: Associa a engenharia social do *hacking underground* e é visceralmente ligada ao livre acesso a informação, a fim de tratá-las, e a sua estrita confidencialidade, para garantir a privacidade dos usuários e impedir as contestações judiciárias de prejuízo a vida privada.

*

[17]Wikinomia, nome dado à economia colaborativa a partir do nome da plataforma colaborativa wiki que por exemplo serve de base a enciclopedia online Wikipedia.

[18]Titulo de um livro recente tendo feito referencia na area da programaçao do web collborativo : SEGERAN, Toby, *Programming Collective Intelligence: Bulding Smart Web 2.0 applications*, O'Reilly, 2007.

Ao observar essa tipologia, há uma *heteroglossia* (BAKHTIN, 1981, 1986), ou seja, uma variedade no mesmo código lingüístico, no qual se acha uma discussão incessante sobre a liberdade. O que constitui o discurso moral dos hackers, e o que diferencia as éticas deles, é a elaboração de um sentido por volta do que é a liberdade e o que significa ser livre. Essa diversidade dá um dinamismo às comunidades hackers, que passam então de uma variedade de discurso a outra, e mudam assim de registro de referências sem se preocupar muito com contradições de conteúdo, de estilo ou de efeito político. Ademais, a trajetória discursiva que cerca a colaboração nas comunidades está em negociação constante, Porém, ao longo dessas negociações, pontos fixos ficam. À defesa da liberdade de informação, que se junta a uma tradição liberal que encontra então uma nova visibilidade e um novo discurso heteróclito em harmonia com a 'era digital'. Entretanto, afirma-se que, o que se junta com esses discursos é uma prática comum da programação. Ou seja, qualquer área tecnológica ou aura política desses tipos ideais designam comunidades que têm com semelhança, como atividade comum, o fato de escrever código. Nesse sentido, Coleman afirma que "a liberdade do software livre, enquanto é influenciada por sensibilidades liberais maiores, é fundamentalmente modelado pelas pragmáticas da programação e o contexto social do uso da internet."[19] (COLEMAN, 2004, p.509).

2.3 As pragmáticas da programação como abordagem para seguir a interação dos usuários desenvolvedores com a informação

Num esforço de definição do ato de programar, Andrew Goffey usa da definição de 'dizer' feita por Michel Foucault em *A arqueologia do saber*: "~~Dizer~~ [programar] é fazer algo – algo diferente de expressar o que alguém pensa, traduzir o que alguém sabe, e algo diferente de brincar com as estruturas da linguagem."[20] (GOFFEY, in FULLER, 2008, p.14). Ao observar os usuários desenvolvedores como programadores, uma dualidade parece dirigir a tarefa

[19] "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."

[20] "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."

deles: por um lado trata-se de um ato muito preciso, um tratamento racional de uma situação racional num contexto racional. Porém, por outro, o discurso entre programadores sobre o objeto deles, a ontologia que o autor desenvolve sobre sua obra, traz muitas referências subjetivas, intersubjetivas, e próxima a expressão de uma natura artística e estética do código (2.3.1). E observando e analisando essas naturezas artísticas e reguladoras do ato de programar, poderemos, então, expor nosso paradigma de análise, isto é, as pragmáticas da programação como vetor do agnosticismo do movimento SL/CA e com relação à informação (2.3.2).

2.3.1 A programação como arte e regulamentação

A programação como arte e a estética do código

"Poeta para poeta. Eu imagino você
na margem da linguagem, no inicio do verão
em Wolfeboro, New Hampshire, escrevendo código.
Você não tem a noção do tempo. Nem a noção dos minutos.
Eles não podem alcançar seu mundo,
seu computador cinza
com quando já agora jamais e uma vez.
Você tinha perdido os outros sete.
Este é o oitavo dia da criação.
...
Estou escrevendo numa tela azul
como qualquer morro, como qualquer lago, compondo isto
para te mostrar como o mundo recomeça:
Uma palavra de cada vez.
De uma mulher para a outra"[21]
Código, BOLAND, 2001.

A Arte da Programação é o título de um conjunto de quatros volumes reconhecidos por ser uma das maiores obras didáticas da área de computa-

[21]Homenagem da poeta Eavan Boland à Sra. Grace Murray Hopper (1906-1988), programadora de dum compilador para a linguagem COBOL, durante os anos 40.“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.”. BOLAND, Eavan, Code, 2001.

ção, e foi nomeada com uma das doze maiores monografias científicas pelo jornal *American Scientist*. A redação desse livro iniciou-se em 1962 quando a editora Addison Wesley propus a Donald E. Knuth, então doutor em matemática de 24 anos e candidato a uma vaga de professor, de escrever um livro sobre compiladores. Até hoje o livro é atualizado com frequência pelo autor, e um quinto volume esta em preparação para 2015. Segundo Maurice J. Black, essa obra segue uma corrente de ensino e comentário dos programas e da programação comparável a um esforço de crítica literária. Antes de *A Arte da Programação*, os *Comentários de Lions sobre a sexta versão de Unix* de John Lions já acompanhavam os programadores no estudo do código fonte do sistema da AT&T, então disponibilizado. Mais do que um estudo, tornava-se a lida de uma entidade estética cujos comentários revelariam os sentidos, particularidades e lógicas (BLACK, 2002).

Além do fato que os escritos sobre código possam ser considerados como crítica literária, o código ele mesmo aparece como obra em si. Esse ponto de vista é perfeitamente encarnado pelas teorias da programação literária (*Literate programming*) desenvolvida por Donald Knuth:

> O praticante da programação literária pode ser olhado como um ensaísta, cuja principal preocupação é expor com uma excelência de estilo. Tal autor, com tesauro nas mãos, escolha os nomes das variáveis com cuidado e explica o que cada umas delas significa. Ele ou ela empenha-se para um programa que é compreensível porque seus conceitos foram introduzidos numa ordem que é a melhor para o entendimento humano, usando uma mistura de métodos formais e informais que gentilmente reforçam-se cada uns os outros.[22]
>
> KNUTH, 1983, p.1.

Assim a analogia feita da programação com a literatura reformula uma metodologia do ato de escrever código. "Segundo Knuth, o melhor código

[22]"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"

não é escrito desde a perspectiva pragmática de um engenheiro, mas sim da perspectiva artística de um autor"[23] (BLACK, 2002). Mais radical ainda no sentido de considerar o ato de codificar como uma prática poética, o movimento de Poesia Perl (*Perl Poetry*) usa a linguagem de programação Perl, caracterizada por o uso de funções designada em termos ingleses "naturais", para traduzir e escrever poemas compiláveis (Figura 2.1, p.47).

Figura 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.
```

A metáfora da arte não é estrangeira aos fatos de linguagens comuns dentro do meio da informática. Como notam Samir Chopra e Scott D. Dexter, a "beleza" do código é um assunto confortável para os engenheiros da computação e a linguagem para descrevê-lo tem muitos adjetivos emotivos. Um conjunto de instruções para o computador é bonito ou feio, limpo ou sujo, leve ou pesado, fantástico, impressionante, horrível, ilegível, etc (CHOPRA e DEXTER, 2007). Nesse sentido, um pedaço do código do kernel Linux de 1990 ilustra (Figura 2.2, p.48):

Embora esteja funcionando corretamente, esse código é descrito como "feio" (*ugly*) por seu próprio autor, por não ser a melhor solução ao problema encontrado, nesse caso sincronizar o horário do sistema operacional com o

[23]"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"

Figura 2.2: Pedaço do 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 )
```

relógio físico do hardware. Poderia acreditar-se que se um código faz funcionar uma ferramenta e que seu resultado é o qual o usuário esta esperando, o código é válido, correto. Porém a relação do programador com seu código, com os outros programadores e com o objeto programado com um tudo – aqui o sistema operacional – exige uma elegância lógica que deve fazer sentido para seus atuantes. Esse código deve regular a interação do software com o hardware, do usuário com ambos desses, e dos desenvolvedores a vir com ele mesmo. Essas interações formam um objeto abstrato sujeito a muitas interpretações e construções diferentes, e como o artista extrai uma obra única do seu corpo ou de uma representação, pode-se considerar "o código como um tentativa da capturar a beleza de um objeto abstrato, o algoritmo." (CHOPRA e DEXTER, 2007, p.77). Comparando com a citação que segue dum crítico literário tentando definir a poesia: "Então! A poesia seria o momento exato, a montagem exata que é obtida para que enfim o real que conseguiu ser delimitado, é restituído graças ao material."[24], um algoritmo por ser uma

[24]"Donc! La poésie serait le moment où le mot exact, l'agencement exact est

montagem única que delimitiu interações e por ser restituída graças a um programa, é, então, como a poesia, uma arte.

O ponto em entender o código como arte é de poder delimitar características do software livre própria a sua natura artística. Sendo que a estética diga algo sobre o objeto software e sua criação, sobre a relação entre o programador e seu artefato, tratar de software livre é delimitar as especificidades de um código aberto nessas interações. O principal argumento das comunidades FOSS nesse sentido é o fato de que, como o artista é influenciado por seu contexto de produção, pelas obras com as quais ele interage e se inspira, o modelo de desenvolvimento colaborativo característico do software livre permite a uma estética muito forte de se estruturar. Como um bom autor, deve-se ler bons escritos, um bom programador deve inspirar-se em bons códigos para realizar as exigências comunitárias de estética. Nesse sentido, e não sem ironia, Bill Gates afirma:

> O melhor jeito de treinar [para ser um bom programador] é de escrever programas, e de estudar excelentes programas que outras pessoas têm escritos... Você deve querer ler o código das outras pessoas, e depois escrever o seu, e depois chamar os outros para revisar seu código. Você deve querer ficar nesse círculo incredível de retorno, onde você encontra pessoas ao nível mundial para lhe dizer o que você esta fazendo errado... Se um dia você fala com um bom programador, você descobrirá que ele conhece suas ferramentas como um artista conhece seus pinceis. E impressionante de ver até que ponto bons programadores tem em comum no jeito deles de desenvolver... Quando você traz essas pessoas a olhar um pedaço de código excelente, você observa uma reação muito, muito comun.[25] (Bill Gates, in LAMMERS, 1989, p.83.)

obtenu pour que enfin le réel qui a pu être cerner, est restituer grâce au materiau !", Fabrice Luchini em dedicace no Virgin Megastore em Paris, 17/11/2008. http://www.dailymotion.com/relevance/search/luchini/video/x7yrj5_fabrice-luchini-en-rencontre-dedica_creation

[25]"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 programmers have in common in they way they develepoed... When you get those

Assim, o software livre facilita uma interação sem restrições com códigos diversificados. A inspiração por outros códigos, além de ser possibilitada, é incentivada por uma tradição de forte crítica e comentários dentro das comunidades FOSS. Há uma criatividade de grupo que caracteriza o produto final como sendo um "melhor" código porque são escritos por melhores programadores, porque eles são formados em contato com esse mesmo código "melhor", formado paulatinamente dentro de uma comunidade com altas exigências estéticas. Ademais, dentro das comunidades FOSS há a idéia de que a exigência tradicional de estética e elegância técnica faz "sobreviver"o que a literatura clássica perdeu. Nesse sentido o Maurice Black comenta:

> [A crítica literária] demonstrou seus compromissos políticos quase abandonando a literatura e a estética como seu tema [...] e, mais ainda, demonstrando sua falta de fé na estética através sua reificação da feiúra ao nível do estilo da critica e de escolha de tema – com o objetivo de transgredir e destabilizar os cânones literários a fins políticos, os teóricos culturais agora estão com alacridade pronta para assuntos como loucura, tortura, amolação, monstruosidade, pornografia, e doença. A cultura informática, por outro, adotou um modelo tradicional de estética literária, como meio de mudança efetiva, achando uma utilidade política e um valor social a um produto bem acabado, que é no mesmo tempo inteiramente operacional e lindo a contemplar como um todo.[26]
>
> BLACK, 2002, p.20.

people to look at great piece of code, you get a very, very common reaction."

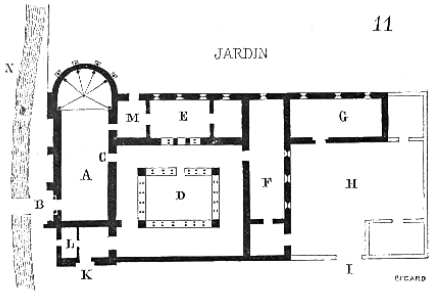
[26]"The first has demonstrated its political commitments by all but abandoning literature and aesthetics as its subject matter [...] and even by demonstrating its loss of faith in aesthetics through its reification of ugliness at the level of critical style and choice of subject matter – with the goal 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."

A programação como arquitetura e a regulação pelo código

Quando num estabelecimento privado de alimentação, alguém volta da sala de banho para a sala principal, como a praça de alimentação de um shopping por exemplo, encontra-se quase sistematicamente uma porta dando acesso às cozinhas, aos armazéns, ou a um lugar de descanso para os funcionários. Colocado na porta, acha-se uma mensagem proibindo o acesso: "acesso restrito", "entrada proibida", "reservado aos funcionários". Assim, a pessoa dirige-se em função de seu estatuto (freguês, funcionário, segurança) pelo caminho que ela pode, para alcançar o lugar que ela quer. Quando alguém se conecta a sua conta Gmail, seu computador lê dados de um disco rigido de um servidor de um centro Google. O navegador dele lê então dados que são fisicamente vizinhos de numerosas outras contas Gmail. Porém, uma regra o impede de direcionar a cabeça de leitura do disco rigido para outra conta, como o tinha feito para a conta dele. E uma regra codificada numa linguagem para máquina que permite essa operação. Tem nesse servidor um sistema operacional que administra contas como espaços e privilégios distintos (usuários diferenciados, administrador de sistema, etc.). Então, como no shopping, uma arquitetura física vem delimitar salas e corredores, cujos acessos são restritos por normas expressas por símbolos, para controlar a atividade humana, o software de um servidor de email vem regular a atividade de seus usuários por uma arquitetura e leis.

Figura 2.3: Comparação entre um plano de arquitetura e um código delimitando espaços discos para usuários

Um plano de arquitetura



Um código Linux padrão delimitando os espaços de vários usuários

```
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
```

Desde a publicação do livro de Lawrence Lessig, Code and others laws of cyberspace (LESSIG, 1999), afirmar que o código é uma lei que regula os comportamentos por coerções e regras tornou-se uma idéia comum dentro das análises da sociedade da informação. Com metáforas detalhadas da lei, da arquitetura física, das normas sociais e dos mercados financeiros, o autor mostrou que o software regula os comportamentos, que "o código é lei". A partir dessa observação, a reflexão de Lessig vai se articular segundo o silogismo seguinte: se as instituições podem regular o software e o software pode regular os indivíduos, então as instituições podem regular os indivíduos pelo software. Assim sublinha-se a importância de entender as implicações políticas e sociais das decisões técnicas, por que elas encarnam uma nova forma de lei. De maneira mais geral também, discute-se as mudanças que os dirigentes devem promover para a Internet.

James Grimmelmann faz uma crítica do trabalho de Lessig e dos debates que nasceram das analogias entre software, lei e arquitetura. Embora essas metáforas sejam consideradas como heurísticas à compreensão dos papéis sociais e políticos do código, Grimmelmann aponta a falta de discussão das diferenças qualitativas entre software e lei, e entre software e arquitetura. Assim, se essas analogias fazem muito sentido para entender o objeto software no seu contexto social e político, várias diferenças de naturezas, de modo de operação diferenciam caracteristicas desses objetos. Para ele, a comparação com a arquitetura permite sublinhar a natura automatizada e imediata do software, enquanto a da lei permite apontar a sua natureza plástica. O software é *automatizado* porque uma vez que ele foi programado, ele age sem precisar de nenhuma intervenção humana. Além da própria norma, que uma vez estabilizada se reproduz com autonomia, algoritmos performáticos, comunemente chamado de 'inteligentes' podem adaptar-se a evolução dos comportamentos que eles regulam. O software é *imediato* pois, em vez de apoiar-se nas bases da sanção, ele simplesmente impede a ação de acontecer. Por um lado a coerção não tem graus diferentes dependendo da sua apreensão pelos comportamentos, ou seja do medo da sanção, pois a ato *não pode* acontecer. Enfim, o software é *plástico* porque é concebível qualquer sistema que um programador possa imaginar e descrever com precisão. Isso revela também a natureza frágil ou quebrável do código, isto é, que qualquer falha que possa ser imaginada pode ser explorada. Nesse contexto, uma característica suplementar do software diferencia o código aberto do fechado. Um software cujo código é aberto é transparente, seus modos de regulaçoes são acessíveis ao entendimento dos sujeitos dos quais ele determina o campo dos

possíveis. Por outro, um software fechado mantém seu algoritmo regulador escondido, com fato único e autônomo, independente do entendimento que seus sujeitos têm dele.

2.3.2 As pragmáticas da programação como paradigma de análise

As pragmáticas da programação: a interpretação sócio-política de um conceito lingüístico

Umas das definições mais antigas da pragmática é feita por Charles W. Morris: "A pragmática é essa parte da semiótica que trata da relação entre signos e os usuários dos signos" (MORRIS, 1938). De um ponto de vista estritamente linguístico, a pragmática trata antes de tudo do sentido, como a semântica, mas para ela, é o uso feito das formas linguísticas que determina o sentido que elas possam ter. Na sua asa mais formalista, notavelmente na obra de Yehosua Bar-Hillel (BAR-HILLEL, 1970), usa-se de paradigmas como o estudo dos símbolos lexicais, do sentido literal e do sentido comunicado, e dos atos de linguagens.

De maneira geral, a pragmática permite questionar vários fatos da linguagem: quando falamos, o que fazemos? O que dizemos exatamente? Quem comunica com quem? Quem sou eu para meu interlocutor? Qual sentido é necessário à coerência de minhas palavras? Uma palavra tem um sentido literal? Quais são os usos da linguagem? (ARMENGAUD, 2007). Assim falar sobre "pragmáticas da programação" seria questionar do mesmo jeito o ato escrever código: o que fazemos quando programamos? O que programamos? Como o usuário percebe o computador? Como o computador recebe as instruções do usuário? Como se formam os erros de interpretação e de sintaxe? Quais são os usos das linguagens, as suas evoluções? E, principalmente, em que medida a realidade do programador e sua capacidade a programar interagem, determinam-se uma com a outra?

Numa abordagem filosófica da pragmática e das interpretações sociais e políticas que ela permite, a pragmática reorientou o olhar da ciência da linguagem sobre os interlocutores. A complexidade deles, mostrada por sua análise, traz o analista a questionar os conceitos de sujeito e indivíduo, notavelmente se focando sobre a interlocução. Assim, se há uma Razão, ela somente pode ser real com a validação intersubjetiva do saber. Aplicada à estética, a abordagem da pragmática faz das obras artísticas um conjunto

de formas simbólicas cujos sentidos e referências dependem das condições de vida e de ação dela, de seu contexto. A arte é então uma experiência que tem por consequência comunicar com um contexto que vem dar a ela seu sentido: "Eu não digo que comunicação é o propósito de um artista. Mas é uma consequência de seu trabalho."[27] (DEWEY, 1934).

Assim sendo, a analogia entre a programação e a arte pelo uso da idéia de estética do código (CHOPRA e DEXTER, 2007) demonstra que, vários paralelos estão aceitos entre escrever código e escrever literatura, entre comentar códigos e fazer crítica literária, entre escrever scripts como se escreve poesia (BLACK, 2002). Por isso, aceitando a idéia de que a programação possa ser considerada como uma das "belas artes" (LEVY, 1992), o conceito de pragmática ajuda a afastar as noções de sujeito da análise do movimento do software livre e focar-se sobre a constituição do corpus de código que ele constitui. As diferentes comunidades e suas interações por volta de projetos de software constituem uma *práxis* que se revela ao observador pela análise das relações desenvolvidas com a informação.

A pragmática da programação como relação à informação e suas políticas informais

A hipótese que esta por traz do uso do paradigma da pragmática como ferramenta de análise do movimento do software livre é que a programação é um discurso sobre a informação. Um programa trata, gera, cria, modela informações, e as pessoas que os criam geram um discurso sobre essas informações, escolhem o jeito que a máquina vai tratá-las. Cada protocolo de rede trata os fluxos de informação de um jeito diferente, e o esforço de configuração feito pelo administrador de uma rede determinada, realiza escolhas específicas que também vêem estruturar a informação, sua difusão e sua recepção. De essas escolhas e regulações que constituam a relação à informação, os programadores constróem uma ontologia que o programa vem automatizar. Nesse sentido, a noção de pragmática permite isolar esta tarefa do programador e desenvolver uma visão ampla das interações que acontecem em torno do ato de programar, considerado como discurso. Além das interações entre os próprios usuários desenvolvedores, podem ser consideradas as interações deles com o objeto computador, suas lógicas, seus limites. Ela mantém uma relação privilegiada, por ser muito normativa, decisiva,

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

com a máquina que trata a informação, então entendido como *dispositivo*, ou seja, "conjunto heterogêneo que resulta do cruzamento das relações de poder e de saber" (AGAMBEN, 2007). Essa noção de "dispositivo"é central ao esforço das comunidades FOSS dentro do movimento tecnológico. A máquina isolada, particular, define-se, nas suas funções e possibilidades, pelos softwares que ela usa para tratar as informações que o usuário joga nela. As normas e estruturas de tais softwares resultem das relações de poder e de saber presentes no processo de codificação do programa. Então a pragmática do programador é a etapa a mais fundamental (após a criação da própria linguagem de programação), onde essas relações realizam-se.

Desta forma, a informação é estruturada pelo objeto software, e as comunidades F/LOSS geram um discurso em cima dos fluxos informativos que geram um conjunto automatizado e plástico que dá seu conteúdo ao SL/CA como movimento político e regulador do espaço digital. Porém, como argumentamos, a pragmática da programação é o vetor do agnosticismo político do movimento SL/CA pelo fato que é a lógica da programação que determina, e permite, a ausência de discurso político aparente, ou reverenciável de maneira comum. Portanto, as pragmáticas da programação formam um discurso próprio sobre a sociedade de informação que constrói uma base de referência própria por seus atuantes. Assim, o usuário-desenvolvedor cria um discurso que é o ato. Com essa idéia, entendemos que o código é uma teoria, uma abstração e uma pragmática, ele é a escritura de um texto, uma prova, e a realização de uma experiência concreta. Por isso, idéias políticas, como a liberdade de expressão, são implementadas pelas comunidades no processo de produção e no objeto produzido. Um exemplo é a tecnologia Wiki, que foi construída em comunidades 'livres' com dinâmicas participativas, para produzir um software livre, permitindo a criação de conteúdo coletivo aberto. A criatividade de grupo cria um processo cujas condições se reproduzem no produto (SAWYER, 2003).

Em suma, o esforço do observador do movimento deve ser de emitir hipóteses ao nível mais fundamental desse discurso – o código – para entender a construção das entidades políticas do movimento SL/CA dentro da tipologia de suas políticas informais (transgressão, civismo, inversão, colaboração).

Capítulo 3

As pragmáticas da Programação e suas lógicas políticas informais

Nós observamos no capítulo anterior as significações culturais do movimento do software livre por meio de: a) da noção de ética hacker, o que permitiu apontar quatros idéias-tipos de lógicas de suas políticas informais (transgressão, civismo, inversão e colaboração); e, b) construímos a idéia de pragmáticas da programação como sendo o vetor da interação do usuário-desenvolvedor FOSS com a informação, e por isso, nos permite emitir hipótese ao nível o mais fundamental da atividade do programador – isto é, o código – para entender a construção da entidade política do movimento SL/CA.

Para esse propósito, propomos aqui uma tipologia de três pragmáticas que se encontram pautadas no discurso das comunidades livres. Em primeiro lugar, consideramos a pragmática de *liberdade* entendida como a ideologia atrás do movimento GNU e os primeiros rastros da cultura do software livre. Por exemplo, pode-se escolher uma ferramenta pelo fato dela estar sob licença GPL. Assim, ela parece como imperativo superior a qualquer outro, enquanto procura-se a construção de uma solução antes de tudo homogeneamente "livre". Ainda, a pragmática da liberdade é o discurso e as interações do movimento SL/CA que privilegia as tecnologias "100% livre", no sentido que o entendem os atuantes.

Além disso, encontramos a idéia de *abertura* nas discussões sobre a compatibilidade das tecnologias, principalmente os protocolos e os formatos de arquivos. A pragmática de abertura seria, então, o discurso dos atuantes do SL/CA sobre essas problemáticas, e o que privilegia essas características sobre as outras.

Por fim, a preocupação com a *segurança* acha-se particularmente nas problemáticas de estabilidade dos sistemas de informação e suas defesas contra eventuais ataques na rede. Assim, a pragmática da segurança seria refletida nos esforços das comunidades em codificar programas antes de tudo seguros, mesmo se as tecnologias produzidas sofrerem a falta de desempenho, de compatibilidade ou se as licenças híbridas protegem parte do código.

Ademais, neste capítulo, o nosso propósito é fazer interagir a tipologia de lógicas das políticas informais próprias à ética hacker com as pragmáticas da programação para que se revele as propriedades políticas do ato que consideramos como fundamental à atividade social de programação. Para seguir esse caminho, trataremos cada uma das pragmáticas isoladas com a apresentação de um estudo de caso, analisado a partir das quatro lógicas políticas. A saber, a pragmática de liberdade será examinada a partir do estudo da comunidade gNewSense (3.2), já a de abertura com o apoio do projeto SAMBA (3.3), e a de segurança por meio dos sistemas BSD (3.4); e finalmente, apresentaremos um quadro de comparação das interações conceituais desejadas (3.5).

3.1 Pesquisa de campo e metodologia

A analise a seguir é o resultado de um trabalho de campo realizado ao longo do meu processo de mestrado, iniciado em julho 2006. O primeiro ato que quero considerar relevante a respeito da minha pesquisa é o uso progressivo de todas as tecnologias livres disponíveis no meu trabalho universitário. Por exemplo, em destaque, a migração do sistema Windows para o sistema Linux (Ubuntu). De fato, tal ação necessita bastante procura de informação na Internet para se familiarizar com um sistema ainda não "intuitivo"como pretendem ser seus concorrentes proprietários. Boa parte dessa procura de informação realiza-se interagindo com outros usuários nas plataformas virtuais de "ajuda"de todas as comunidades. Construiu-se, então, um vocabulário específico permitindo expressar os problemas encontrados ao longo do nosso caminho. Esse próprio vocabulário é aquele mesmo que permite interagir com as comunidades para assuntos mais *off-topic* (fora do assunto) e oferece então a entender o ambiente político-ideologico que se encontra ao redor das comunidades livres para justificar as suas escolhas de usar ou não software livre.

Tendo adquirido um bom nível técnico, suficiente para ser “autônomo” e somente precisar de um acesso ao Google e/ou ao Wikipédia para conse-

guir solucionar a maioria dos problemas encontrados, aproveitei um contato pessoal para entrar em um centro de computação de um departamento de ciências exata[1] de uma grande universidade paulista. Assumi então a posição de voluntário com carga de técnico de apoio, isto durante cinco meses (de maio até setembro 2008). Rapidamente foram-me atribuídos projetos precisos: atualização das estações Linux-Fedora da versão 7.0 para a versão 9.0, preparação da distribuição Ubuntu para a padronização e implementação dentro do departamento (substituição a Fedora), alguns testes para a configuração de um servidor de virtualização de sistemas para ajudar o atendimento (Xen, VirtualBox). Além dessas tarefas predeterminadas, eu tinha que assumir vários serviços de atendimento ao usuário, isto é resolver problemas diários que alguns pesquisadores e alunos do departamento encontravam ao usar ferramentas livres.

Embora o centro usasse de tecnologias livres e tecnologias proprietárias, podemos observar que interagi principalmente ao redor de tecnologias de código aberto. Isso aconteceu pelo fato de eu me apresentar aos administradores do centro de computação como interessado especificamente nessas tecnologias, por causa da minha pesquisa, e que isso justificava meu estatuto de "voluntário"[2]. Porém, embora o fato de ser voluntário me deu mais flexibilidade na organização de meu trabalho, assumi uma carga horaria normal, ou seja, 8 horas por dia, 5 dias por semana.

O meu processo de aceitação dentro do centro de computação foi facilitado pelo fato que minha imagem de aluno francês ultrapassava muito aquela de cientista social. Em relação aos funcionários, a intergração foi mais rápida ainda, sendo que eles se interessavam mais na qualidade do meu trabalho e de minhas observações do que em qualquer outra coisa. Em realção aos estagiarios (mais jovens), como disse, eles eram mais interessado em detalhar seus preconceitos e fofocas sobre à França (Moulin Rouge, Zidane, Carla Bruni, Sarkozy, etc.) do que qualquer outra coisa. Não encontrei nenhum problema para fazer as perguntas que eu queria, e assim, encontrar todas as informações que procurava. Atribuo esta facilidade com minha entrada em campo a dois fatores 1) tentei ser amigável com todos e 2) competente no

[1]Parece-me relevante precisar a área desse departamento (ciências exatas) porque esses departamentos fazem, de maneira geral, um uso muito mais intensivo das tecnologias da informação do que os departamentos afiliados à apelação "ciências humanas".

[2]concretamente, isto quer dizer que recebi nenhuma verba por esses serviços, e que a responsabilidade jurídica da minha presença aqui dependia do meu departamento (DCP/IFCH, UNICAMP) e não de esse centro de computação.

meu trabalho.

Entre as perguntas que eu fazia, bastante eram de ordem históricas, ou seja, eu pedia tanto do que possível como tal ou tal tecnologia nasceu, evoluiu, superou seus equivalentes, etc. Também eu tomava cuidado em entender os conceitos modernos de computação (protocolos, termos organizacionais, etc) através de um olhar mais "antigo"para compreender suas raizes[3]. Nesse sentido, as perguntas que me serviram mais foram "como era antes?", "como chegou a ser assim?"e "para onde vai?"

Basicamente, esse trabalho me permitiu sistematizar o contato que eu tinha com as comunidades livre e sair do "romantismo" que pode sofrer uma interação sob propósitos pessoais. Em vez de encontrar um problema especifico para minha situação especifica, eu aprendi procurar de maneira sistemática soluções para uso amplo, isto é, para as centenas de computadores em uso no departamento. As relações profissionalizam-se dentro dos fóruns de ajuda e os diálogos ficam estritamente ao redor das problemáticas técnicas. Desse jeito trazem-se os discursos encontrados dentro do próprio centro de computação e os debates internos vêm realizar as escolhas técnicas ainda em duvida na plataforma virtual. Esta observação me levou a formar a hipótese que acabaria sendo central na minha dissertação sobre a pragmática da programação. De fato, observei que os critérios selecionados pelos atuantes durante esses debates eram técnicos, justificava-se tal ou tal opção por argumentos técnicos, porém era nesse momento que se formulava a aura ideológico-política dos projetos mencionados. Ilustramos com um debate sobre as distribuições do Linux:

> **Eu**: Então olhei mais ou menos todas as ferramentas disponíveis no Ubuntu e acho que é possível começar fazer os testes
> **1**: bom, acabamos o a padronização do Fedora, e depois passamos nos testes do Ubuntu. Acho que vai ser mais simples para a parte multimídia. Fedora da um trabalho para os plug-ins...
> **Eu**: Pois é... Com Ubuntu fica muito simples, permite instalá-los com um comando só
> **1**: Tem idéia de porque eles não fazem a mesma coisa do que o Ubuntu? O Fedora esta perdendo usuários com essa atitude...
> **Eu**: o Ubuntu quer facilitar o acesso para os usuários, o Fedora é mais fiel a respeitar as licenças dos softwares disponibilizados.
> **2** (*ouvindo o argumento*): Seria você, trabalharia em cima do Kurumi, ele já faz tudo isto, DVD, flash, Java, tudo tá disponho.
> **1**: mas ai vai dar problema para os aplicativos científicos, ele não é feito para

[3]exemplos ilustrativos: *Internet* é uma rede feita de protocolos, um *Podcast* é um vídeo

isso. Não, acho que Ubuntu já ta bom.
2: instalei ArchLinux ontem na minha maquina para experimentar... Muito legal esses sistema, bom não da para colocar aqui, as atualizações são demais rápidas. Mas eles tentam organizar o sistema que nem o BSD, bem logicamente, para os desenvolvedores mesmo.
...
1: pois BSD é um saco viu, estou trabalhando com o servidor de impressão e para fazer dialogar ele com o Linux dá um trabalho. Já no outro no [departamento vizinho] eles homogeneizaram todas as máquinas para facilitar o serviço... Mas aqui o povo quer guardar o BSD. Ta é mais segura, estável, etc.

Esses debates freqüentes aconteciam também com os usuários universitários. Porém percebia-se uma certa confusão entre os argumentos ideologico-politicos a favor do SL/CA e, ao mesmo tempo, uma crítica aos funcionairos técnicos por não disponibilizar as ferramentas fechadas.

Não realizei durante esse campo entrevistas formais, mas usei das interações informais que eu podia analisar depois com os rastos que deixava delas no meu caderno de campo. O dialogo transcrito a cima é um exemplo de rastros que podia-se encontrar no meu caderno. Também, anotei quadros comparativos de tecnologias equivalentes em debates, com os argumentos sublinhados pelos actantes (dentro ou fora do centro de computação), palavras com duplo sentido, piadas especificas às personas que encontrei, etc.

Durante o período de escrita (6 meses), passava várias horas por dia nos canais IRC, olhando e participando de debates técnicos que, de vez em quando, acabavam em debate sobre os propósitos e compromissos sociais e políticos do SL/CA. Entre outros, os canais IRC nos quais participei foram: Samba, gNewSense, FreeBSD, LaTeX, Ubuntu, Fedora, OpenOffice, Hack, Kubuntu, GNU, FSF[4]. Essa interação em tempo real permitiu reproduzir alguns debates presentes no departamento no qual trabalhei, mas numa praça pública maior, pois centenas de usuários conversam aqui na mesma interface. Essas observaçoes ineragiam diretamente com meu processo de escrita, permitindo verificar informações na hora que eu as escrevia, jogar debates dentro dos canais quando me faltava opiniões formalizados. Isto permitiu tornar previzível a maioria dos pontos de vista sobre os assuntos que eu tratava, o que, eu acho, é um dos passos anterior e necessário à sistematização. Tentativas de análises quantitativas de dados qualitativos foram realisadas em cima dos arquivos IRC das comunidades Debian e Ubuntu, principalemente

[4]servidor: irc.freenode.net:6667

a respeito do uso das palavras "livre"e "aberto". Porém, nao encontrei resultado satisfatórios sendo que não tinha ainda achado tipologias interessantes para explorar.

No IRC, obteve-se uma interação completa sobre o assunto desejado com duas pessoas, conscientes do meu propósito universitário. Neste caso podemos falar de entrevistas formais que guardei e analisei na sua integralidade. Aconteceram dentro da comunidade gNewSense, e uma parte é transcrita no ponto 3.5.

De maneira geral, o que é importante sublinhar é a grande abundância de informação dentro das comunidades livres. Como nota o antropólogo Christopher Kelty, "*Geeks talk a lot*"(KELTY, 2008) e eles como eu sendo interessados no assunto, não tem retenção de informação nenhuma. Que seja dentro do centro de computação onde eu fiz campo, ou nas inumeráveis plataformas virtuais nas quais interagi, as minhas perguntas formulavam-se seguindo os fluxos presentes de interesses e não respondiam a um roteiro pré-estabelecido, do tipo questoniário ou linhas diretivas. Por um lado isto me impede de reformular na presente seção o caminho exato e detalhado da minha pesquisa de campo e da minha percepção teórica. Por outro lado, quero pensar que tal método permite, no prazo curto de um mestrado, mergulhar mais profundentemente nas representações dos atores que foram encontrados. Iniciei meu processo de observação sendo chamado pela grande versatilidade desse movimento, dos discursos e posicionamentos político-econômicos que ele permite, e tentei sistematizar essa diversidade juntando os fluxos discursivos que encontrei.

3.2 As pragmáticas de liberdade: A comunidade gNewSense

> Software não-livre nunca é uma solução, então, por favor, não racionalize, justifique, ou minimize as consequências de propor software não-livre como uma solução.[5]
>
> 5a diretriz comunitária de gNS

[5]"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"

A pragmática de liberdade é a primeira que aparece em contato com as comunidades SL/CA. Por isso, os responsáveis quererem defender por meio da tecnologia o direito a participar da evolução livremente, pois o código aberto é a condição de uma idéia de liberdade. O modelo GNU e o domínio de código sob licença GPL assumem de maneira radical esse propósito do movimento SL/CA. Tais pretensões foram abraçadas emblematicamente pela comunidade gNewSense, que tenta construir uma versão 100% livre de Linux, pois o mesmo no é "livre"no sentido que quer promover a FSF e as comunidades perto dela. As distribuições de Linux apontadas sempre permitiram a instalação de software com restrições que, no final, constituíam um sistema sob direitos heterogêneos. Essa banalização de códigos restritos no ambiente Linux fez um passo pela frente com a adoção em 2006 de pedaços (*blobs*) de códigos fechados no próprio cerne do sistema. Esses blobs foram então incorporados aos sistemas distribuídos, incluindo aos mais dedicados aos ideais do Software Livre, como é o caso da comunidade Debian[6].

Nesse contexto, Richard Stallman, fundador da FSF, e Mark Shuttleworth, patrocinador de Ubuntu, sublinharam juntos a importância de manter um esforço para construir e manter uma versão literalmente "livre"do sistema. Um estudioso, Paul O'Malley, trouxe o debate para ao público através do IRC e, junto com Brian Brazil, começaram a desenvolver uma arquitetura de sistema renovada. Os blobs e os repositórios de softwares fechados e/ou restritos foram tirados e formou-se uma comunidade por volta desse sistema experimental que ganhou o nome de gNewSense (gNS). Os usuários-desenvolvedores confrontaram-se a dificuldades técnicas importantes, tornando o uso do sistema difícil para os propósitos comum e raramente compatível com os hardwares. Porém, atualmente, em Junho 2009, o sistema está disponibilizado na versão 2.2 e conseguiu combinar todas as alternativas possíveis aos programas restritos, oferecendo uma alternativa possível, embora seja ainda complicada, aos compromissos feitos pelas distribuições concorrentes.

Cabe ressaltar que, as análises propostas aqui foram realizadas a partir de materiais encontrados principalmente no Fórum, nas mailing-lists da comunidade[7], nas interações no seu canal IRC[8]. Além da observação dos debates ao vivo, e nos arquivos de anais de chat, foram realizadas entrevistas

[6]Voto da comunidade Debian a favor do uso dos *blobs*: dia 15/10/2006, http://www.debian.org/vote/2006/vote_007.en.html

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

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

de vários usuários-desenvolvedores, entre os quais, foi um dos fundadores da distribuição, Paul O'Malley.

3.2.1 Lógicas de transgressões

As pragmáticas de programação desenvolvidas pelos desenvolvedores dessa comunidade não são entendidas como a transgressão. O hacking, compreendido no seu aspecto underground, não é um vetor desenvolvido no esforço tecnológico que conduz a construção desse sistema alternativo. De fato, os usuários-desenvolvedores gostam de se apresentar, como os do projeto GNU ou próximo à FSF, como 'gnu hacker'. Por ser um ato profundamente político e cívico, as características das pragmáticas da comunidade gNewSense devem ser procuradas nas outras lógicas que constituam nossa tipologia.

3.2.2 Logicas de civismo

As pragmáticas do civismo aparecem dentro de gNS como um conjunto de escolhas para se conseguir um resultado de propósito geral. De fato, as restrições técnicas que se impõem aos hackers da gNS para desenvolver o sistema em cima de uma plataforma com limitações importantes é um ato positivo no sentido de ignorar as alternativas e os meios-termos realizados pelas outras distribuições do Linux. Há um recurso categórico das perspectivas de desenvolvimento 'reformistas' que encaram a mudança para o 'livre' como um processo progressivo que necessita a utilização de ferramentas com licença restrita e/ou próprias. Essa posição dos desenvolvedores reafirma-se cotidianamente nas plataformas de interação com os usuários que peçam ajuda para o uso do sistema e sempre enunciam soluções não-livres. Vejamos a seguir o comentário de um usuário, o mesmo foi escrito para convencer uma pessoa que queria deixar de usar gNS e de experimentar outras distribuições:

> *Instale ubuntu, [...] depois você poderia seguir a evolução do software livre e de gNS, mudar para alternativas livres, um bit após o outro... ok, ubuntu não é livre, mas é mais livre do que Windows.*
> Eu entendo o que você tenta fazer, porém, a mensagem que você manda mesmo é: "tudo bem em usar software não livres por enquanto; quando o software livre não tiver mais diferenciais fun-

> cionais com o não-livre, então pode voltar". Isso torna artificial a ética do software livre e não leva agente em lugar nenhum.[9]

Nesse exemplo, o desenvolvedor cobra o usuário de propor um software não-livre (uma distribuição mais amigável do Linux) como jeito de conseguir chegar até um uso exclusivo do Livre. Isso está na contra mão do esforço cívico da comunidade em agir radicalmente. Nesse sentido, a comunidade gNS se caracteriza por uma forte postura política e por critérios tradicionais, ou seja, um ideal realizado por uma ação comum contra um inimigo declarado. De fato, os quatro tipos de liberdades e a filosofia original do projeto GNU são as bases desse movimento tecnológico de criação de um sistema operacional e plataforma de desenvolvimento de software livre. Isto se dá para manter uma alternativa contra todas as formas não-livres de códigos. Entretanto, para esse propósito, um "sacrifício"técnico de não se poder usar ferramentas básicas do mundo virtual contemporâneo. Por exemplo, assistir aos vídeos online ou acessar sites web "dinâmicos", ou ainda o acesso a vários formatos básicos dos aplicativos de escritório são ações dificultadas. Nesse sentido, os integrantes desse projeto pretendem ativar um ato de civismo, através de pragmáticas de liberdade radical ao facilitar o custo técnico do desenvolvimento de uma alternativa inteira que qualquer uma poderá aproveitar.

3.2.3 Lógicas de inversão

O sistema de propriedade intelectual que estrutura o mundo do software e seus conflitos é baseado em regulamentos, leis, convenções nacionais e internacionais (por exemplo o DMCA – Digital Millenium Copyright Act), que as licenças particulares convocam. Uma tecnologia é fechada ou restrita porque sua licença chama o conceito de copyright ou de direito autoral na licença, caso o usuário concorde em instalar o programa. O propósito do projeto GNU e da licença GPL foi de usar da mesma legitimidade jurídica para defender o oposto, ou seja, a não-apropriabilidade do código disponibilizado.

[9]" 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"

Isto é uma lógica de inversão das defesas proprietárias para constituir defesas do "livre".

Dessa forma, as preocupações legais são no centro da atividade da comunidade gNewSense e por isso, assumem-se os próprios desenvolvedores dos 4freedom checkers, além das categorias tradicionais (codificadores, suporte ao usuário, gerenciamento dos bugs). Esses desenvolvedores monitorão o sistema e todos os programas disponíveis para verificar a ausência de falha legal, permitindo um código de licença restrita a ser instalado. Para isso, um construtor (builder) específico foi programado para testar os códigos instalados e suas compatibilidades com as exigências de liberdade da comunidade. Contudo, a exigência de uma licença compatível com a GPL não é suficiente para esse propósito, e os desenvolvedores procuram estabelecer se os programas não apontam para fontes secundárias restritas. Mais uma vez, as implicações de tal dedicação aparecem particularmente no contato com os usuários; como uma resposta de um desenvolvedor a um usuário, que queria instalar um emulador Windows (Wine, sob licença GPL) para poder utilizar seus jogos (proprietários):

> Um ponto central para entender gNewSense e todo o Software Livre, é de entender que é seu propósito legal de ser completamente software livre. Enquanto preciso pesquisar mais para saber se Wine é livre ou não, o software livre não é para dar um jeito de graça para rodar todos seus aplicativos Windows. É um movimento político, social e tecnológico para um software livre.[10]

A propósito do que constitui a base política e social que evoca o desenvolvedor, apontamos a procura de uma homogeneidade jurídica inédita e absoluta. Para tanto, a maioria dos debates nas mailinglists de desenvolvimento da distribuição estão focados sobre os aspectos legais dos softwares, enquanto os aspectos técnicos, quando não envolvem uma tecnologia alternativa, são deixados à distribuição-mãe do sistema (Ubuntu). Separar o que é ou não livre é a principal lógica da organização do sistema gNewSense. As lógicas de liberdade levam os programadores a escolher quais softwares

[10]"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"

podem ser instalados e quais não podem. Assim, ilustramos uma inversão do radicalismo proprietário para um radicalismo do 'livre', e ainda uma "ignorância" deliberada de todos os 'meios-termos' que se encontram no meio tecnologico do Open-Source.

3.2.4 Lógicas de colaboração

Um espaço de colaboração sem restrições é o propósito final da GPL; e as possibilidades de colaboração instauradas pela comunidade gNS são ao mesmo tempo absolutas e fechadas. De fato, por um lado, o potencial de dinâmica participativa dentro do sistema é infinito, porque todo seu código é livre, acessível, aberto, modificável nos termos da GPL e as suas licenças são semelhantes. Por outro, o espaço criado noa aproveite de outros projetos cujas condições de apropriabilidade são distintas. Por isto, considerando a extrema variedade das naturezas jurídicas dos códigos no desenvolvimento das tecnologias, o domínio GPL e assimilados, são totalmente hermético a boa parte das inovações. Elas ainda devem ser recodificadas (quando não são patenteadas) num projeto que responda aos critérios da comunidade.

Ademais, a ideologia do projeto a respeito das possibilidades de colaboração pertence a dois prazos temporais. Desse modo, entendemos que à curto prazo, há um sacrifício em nome de uma utopia, uma idéia de liberdade absolutamente não restrita que bloqueia as possibilidades de colaboração e, assim, permite as trocas de códigos somente dentro do domínio da licença GPL. Esse compromisso dedica-se inteiramente ao ideal do projeto GNU, para que à longo prazo seja realizado sua concepção de colaboração, absoluta e não restrita. Isto representa um posicionamento realmente ativista dentro das comunidades livres, que volta à heterogeneidade de licenciamento. Meios e fins se misturam para um propósito do movimento, refletindo a ideologia proposta e por isso a contradição que aponta a falta de abertura como restrição a colaboração não atinge as representações dos atuantes voltados a um propósito "maior".

3.3 As pragmáticas de abertura: A comunidade Samba

Umas das principais dificuldades encontradas pelo sistema Linux dentro de instituições foi sua falta de compatibilidade com os protocolos de códigos

fechados. De fato, com a multiplicação das estações Windows, todos passaram a usar os protocolos de redes internas próprias a esse sistema para os computadores comunicarem-se dentro de uma mesma instituição. Em tal ambiente foi difícil para os sistemas Linux, servidores ou clientes gerenciar tais protocolos, ou mesmo, simplesmente comunicar-se com eles. Foi com a intenção atenuar essa carência específica que foi criado o projeto Samba, um projeto agora antigo e com muito sucesso no mundo do SL/CA.

Iniciado como projeto de doutorado por um aluno australiano, Andrew Tridgell, em 1991, o software Samba permitiu pouco a pouco aos sistemas baseados no Unix (Linux, BSD, e outros) de dialogar nas redes com os sistemas e servidores Windows. Na prática, permitiu, dentro de outras coisas, aos clientes windows de interagir com servidores baseados na tecnologia Unix e, inversamente, para os serviços de impressão e de compartilhamento de arquivos. Licenciado sob GPL, o projeto se tornou padrão em todos os ramos do SL/CA e para gerenciar os protocolos de redes híbridas.

Vale à pena dizer que as análises propostas aqui estão realizadas a partir de materiais encontrados principalmente no Fórum[11] e as mailing-lists da comunidade e nas interações no seu principal canal IRC[12]

3.3.1 Lógicas de transgressões

As lógicas de transgressão estão presentes na comunidade Samba pelo simples fato de serem as que fundaram o projeto. Para conseguir se comunicar com os protocolos de rede da Microsoft, os quais são fechados, foi preciso usar os métodos da engenharia reversa. Sendo fechados, os protocolos apresentam-se ao desenvolvedor como uma 'caixa preta', cujas características podem ser descobertas somente pela análise de suas entradas e saídas. Com o uso de packet sniffer, os desenvolvedores usam de ferramentas comuns da exploração de rede para 'sentir' o comportamento dos protocolos fechados e assim determinar qual código permitirá se comunicar com eles. O Samba explora na verdade um dos casos onde tal prática é permitida, isto é, quando é realizada a fins de permitir uma interoperabilidade entre padrões diferentes.

Tal atividade ilustra-se pela exigencia de uma competência técnica avançada, e representa um desafio atraente para seus atuantes: derrubar um segredo proprietário e fazê-lo funcionar com uma tecnologia livre. As referên-

[11] http://www.nabble.com/Samba-f13150.html
[12] #samba@irc.freenode.net:6667

cias comuns dentro do hacking underground estão presentes no discurso dos desenvolvedores à medida que não ultrapassam os limites que a lei permite, pois, por ser um projeto aberto e muito usado, as interações dos usuários-desenvolvedores acontecem à vista de todos.

Porém, num contexto mais geral de observação, podemos revelar aqui umas das características das políticas informais de transgressão, isto é, o ato transgressivo é muito dependente do objeto que ele tenta derrubar. Se nós consideramos o ato de permitir os protocolos livres de dialogar com os proprietários como um ato em favor do "livre", então o ato de transgressão realizado nesse propósito fica necessário enquanto os protocolos são mantidos. Nesse sentido, se as motivações dos usuários-desenvolvedores que participam da comunidade Samba são, por parte, inspiradas pelas lógicas de transgressão que sua codificação necessita, elas estão dependentes da existência desses mesmos protocolos proprietários. Por isso a lógica transgressiva permite tanto aos protocolos proprietários do que é livre de sobreviver num mesmo ecossistema que não é entendido de maneira reivindicatória. A pragmática de abertura na sua lógica transgressiva não sublinha a presença de códigos proprietários na rede, mas procure um 'direito' a se comunicar com eles. O ambiente maior onde estão interagindo esses códigos não é apontado como livre, mas como sendo naturalmente heterogêneo.

3.3.2 Lógicas de civismo

Os métodos de engenharia reversa se encontram em vários projetos livres. Trata-se de 'correr atrás' de formatos de arquivos ou protocolos indispensáveis ao funcionamento básico de um sistema. Como samba procura integrar os protocolos da microsoft ao mundo Unix, open office procure "correr atrás"do formato .doc e Wine atrás das interfaces de programação de aplicativos (API) do sistema Windows para eles serem executados em cima de uma plataforma diferente. Por isso, o resultado que se apresenta ao usuário final não é uma alternativa como é o caso para gNS, mas uma interoperabilidade, um código que permita as tecnologias híbridas de comunicar. Sendo que o propósito mais fundamental das tecnologias da informação é a comunicação, tal propósito tem um impacto muito maior do que a criação de uma alternativa total. Isto é permitir os "meios-termos".

Porém, há uma proposta política forte atrás do uso da engenharia reversa, porque ela é sempre vista por seus atuantes por ser um modelo de desenvolvimento muito fundamental à inteligência humana. Quer dizer que

olhar para um objeto desconhecido e entendê-lo a partir do que ele produz e relacioná-lo com o que foi colocado nele. Essa idéia é como um "direito fundamental"do programador, como do ser humano em geral. Não ter autorização para usar essa forma inteligente é percebido como restringir uma cognição muito natural da cognição. Nesse sentido, um desenvolvedor web francês comenta:

> Bom, eu sou programador web, e trabalho muito em cima da tecnologia Flash que é pouco livre [...] O que eu gosto na filosofia do livre e que ela parece defender meu modo de aprendizagem contra um modelo que parece o proibir [...] Eu nunca fui na universidade, aprendi tudo sozinho, e por isso, tenho falta de conceitos gerais ou fundamentais em engenharia da computação. O que eu sei fazer é ver um negocio e entender como funciona, tanto faz que seja livre ou não, a caixa preta é meu quotidiano.. assim... ha um "bixo" que faz coisas e recebe outras, como ele funciona? Como eu faço para ele tratar uma coisa ou produzir outra ? é que nem bêbê que aprende a falar, ou adolescente que aprende a trepar: isso funciona ? Sim, guardo. Isto não? jogo fora. Isto mais ou menos? preciso dar uma olhada, etc... e assim vou em frente, junto soluções, concertos para desenvolver novos saberes, ferramentas, etc...

Se a interdição de tal prática, um dia, for banalizada ao mundo da computação, há a idéia que a cognição a mais básica do usuário-desenvolvedor nem poderia mais exprimir-se, no sentido de usar seu raciocínio. Por isso, o uso da engenharia reversa é percebido por seus usuários como o fato de 'pensar' um software da maneira mais fundamental. Por enquanto o DMCA considera esta prática como aceitável por fins de observação ou educação. Porém os atores proprietários sublinham o fato que muitas operações de pirataria realizadas com software (cracking, por exemplo) envolvem essas técnicas, e por isso proibi-la ou regulá-la severamente seria um ato eficiente contra seu uso ilegal.

Assim, as comunidades livres como Samba, que praticam esse modelo de desenvolvimento defendem profundamente esse direito apontando seu valor social e político de defesa como uma modalidade primaria do entendimento humano.

3.3.3 Lógicas de inversão

Por não ser uma alternativa, mas um esforço no sentido de incorporar as realizações do mundo proprietário, a comunidade Samba não opera uma lógica de inversão além do que a licença GPL já realiza. Por isso, as outras lógicas aqui examinadas devem ser consideradas de maior importância que essa, para entender as pragmáticas de abertura.

3.3.4 Lógicas de colaboração

Por ser uma comunidade muito aberta às contribuições, tendo usuários desenvolvedores dos mundos livres e proprietários dentro dela, a comunidade samba encarna muito bem as lógicas extremas de colaboração que se encontram no mundo open-source. O foco das pragmáticas de abertura nas suas lógicas cívicas, a interoperabilidade das tecnologias, as lógicas de colaboração aparecem como o último propósito de uma comunidade como Samba. Além disso, a porta de entrada dada às tecnologias privadas para comunicarem com as abertas é uma realização que pode ser criticada do ponto de vista da filosofia GNU-GPL, mas que é muito incentivado na asa pragmática do movimento SL/CA.

Atrás disso, acha-se uma interpretação diferente dos limites entre o que pode permanecer fechado é o que deve ser aberto. A diferença aparece em um nível técnico, enquanto a filosofia GNU procura um 'tudo' livre, ou seja, um projeto como Samba procura uma 'possibilidade' de livre, isto é a oportunidade com um código livre de comunicar como qualquer coisa. O domínio do livre é então entendido como um tudo homogêneo que possa dialogar com outros conjuntos de naturezas diferentes, e por isso ver sua pragmática de abertura acrescentada. Um exemplo relatado por um dos fundadores do projeto Samba trata de outra preocupação atual do movimento SL/CA, que é oferecer uma alternativa livre aos protocolos proprietários de VoIP (Skype, Google Talk, entre outros): "Eu espero só que eventualmente os desenvolvedores de software livre conseguem furtar na rede do Skype e inter operar com os protocolos Skype. Finalmente, há bons antecedentes para isso com outros softwares livres..."[13] (ALLISON, 2005).

Como podemos observar a referência aponta a fraqueza do projeto Ekiga,

[13]"I just hope that eventually Free Software developers can work out how to hook into the Skype network and inter-operate with the Skype protocols, after all, there are good precedents for that with other Free Software..."

alternativa livre ao Skype mal sucedida, o software o mais usado para o VoIP. Enquanto Ekiga oferece um software livre para usar seus próprios protocolos livres nos seus servidores comerciais, Jeremy Allsion propõe como solução o uso de engenharia reversa para permitir um software livre interagir com o protocolo proprietário do Skype. Vemos então que as lógicas de colaboração desenvolvidas pelas pragmáticas de abertura como se acham na comunidade promovem não um meio-termo ou uma concessão do mundo livre ao proprietário, mas uma reinterpretação do domínio do livre a respeito dos protocolos de comunicação.

3.4 As pragmáticas de segurança: As comunidades BSD

As pragmáticas de segurança ilustram um foco especial dado ao desempenho de uma tecnologia computacional. De fato, dentro das preocupações de um programador, como a velocidade, interoperabilidade ou a leveza de uma tecnologia, existe também a noção de segurança. Um software, como todo 'sistema', é quebrável e tem falhas que são oportunidades para uma pessoa mau-intencionada explorar e modificar um comportamento sem se submeter ao algoritmo central. Com a difusão, em grande escala da Internet a partir dos anos 90, sublinhou-se a importância de tais assuntos de segurança para limitar o espalhamento de vírus e as explorações de redes privadas.

De fato, as tecnologias livres sempre revindicaram ser mais seguras por ser justamente abertas, isto é, que qualquer 'falha' fica mais aparente e por isso é mais rapidamente comunicada e consertada. Independentemente das pretensões do livre, e mais por desafio tecnológico, os sistemas Unix baseado nas tecnologias BSD sempre desenvolveram uma preocupação maior a respeito da segurança e estabilidade, particularmente para operar em servidores.

As comunidades BSD são as que se formaram nos arredores da distribuição de Unix desenvolvida na Universidade de Berkeley (BSD – Berkeley Software Distribution) entre 1977 e 1995. Após essa data, dividiram-se em três famílias principais, FreeBSD, netBSD e OpenBSD. Nos focaremos principalmente à comunidade FreeBSD, sendo a mais comum (77% dos sistemas BSD em uso em 2005[14]) e à do projeto OpenBSD por ter projetos conexos mais produtivos (openSSH, openSSL).

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

Mais uma vez, reforçamos aqui que as análises propostas são baseadas em materiais encontrados principalmente no Forum[15] e as mailing-lists da comunidade e nas interações no seu principal canal IRC[16].

3.4.1 Lógicas de transgressões

A estabilidade que as tecnologias BSD disponibiliza aos seus usuários necessita uma intensa atuação no campo da segurança. Por isso, muitos desenvolvedores estão vinculados aos mesmos circuitos de informação que disponibilizam as falhas de sistemas. A lógica profunda da segurança na informática responde a uma troca entre atacantes e defensores.

De um ponto de vista representativo, os atacantes querem achar falhas para explorá-las, mas querem um sistema seguro para usar e por isso procuram. Os defensores querem também achar as falhas para consertar os sistemas e por isso procuram dialogar com os atacantes. Em resumo, cada um dos partidos procura o que o outro acha, e por isso, a alta exigência de segurança dos sistemas BSD aproxima seus desenvolvedores dos campos virtuais do hacking underground.

Porém, o propósito que motiva as atuações no campo da segurança dos desenvolvedores BSD é a segurança do sistema, por isso a transgressão das tecnologias aparece como um meio indireto de realizar este objetivo.

A inversão operada no nível das lógicas de transgressão é a base do processo de tornar seguro das tecnologias BSD. O ato de transgressão que necessita buscar falhas e reproduzir integralmente pelos desenvolvedores, mas o propósito é inverso por ser destinada a correção falha. Um exemplo ilustrativo é o processo de 'auditoria' (OpenBSD code auditing) realizado pela comunidade OpenBSD em todos os software disponibilizados com a distribuição. Tal processo exige de um desenvolvedor que leia o código, procurando possiveis explorações, o que é exatamente o processo que realiza um cracker mal-intencionado para poder explorar um código. Isto vai além da simples relação de compartilhamento de informação em plataformas onde as falhas estão publicadas, trata-se, contudo, da exploração-segura para escapar da colaboração com os crackers. Nesse sentido, a comunidade openBSD argumenta que, com tal modo de desenvolvimento ela conseguiu desenvolver um sistema que não sofreu falhas de segurança durante 5 anos[17] de uso. Tal pre-

[15] http://forums.freebsd.org/

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

[17] Até junho 2002 o slogan de openBSD era: "Five years without a remote hole in the

ocupação determina por grande parte as lógicas de civismo (contribuições) e de colaboração das pragmáticas de segurança das comunidades BSD.

3.4.2 Lógicas de civismo

A diversificação das atividades na Internet, particularmente as envolvendo troca de dinheiro, como é o caso das compras online, ou as consultas de dados sensíveis (conta bancária) foi permitido paulatinamente pelo uso de tecnologias de criptografia em grande escala. Umas das maiores contribuições na área foi realizada a partir da comunidade OpenBSD, cujos projetos OpenSSL e OpenSSH "liberaram"os primeiros esforços nesse sentido. De fato, o projeto openSSH é adotado em todas as distribuições de Unix, ou seja em quase todos os sistemas que não são Windows, e permite a dois sistemas de dialogar inteiramente com uma conexão protegida e criptográfica. Porém, o projeto com contribuição muito maior foi OpenSSL, que pretende oferecer a mesma segurança para as conexões via web. Sem dúvida, muitas das conexões que acontecem na web necessitam um tipo de proteção, como é o caso das compras ou divulgação de senha, para que os dados assim transferidos não apareçam decifrados a quem "lê"o tráfego da rede.

Há uma dimensão política e social a tais atos, próxima aos primeiros passos da criptografia. A segurança e a privacidade estão consideradas pelos atuantes dessa comunidade como uma necessidade social, uma carência. O código assim escrito é um ato político e social no sentido de cobrir essa necessidade. Tal esforço é aproveitado por toda a sociedade conectada. Uma comunicação segura o comércio online, as transações financeiras, a designada "pirataria"de conteúdo intelectual, ou seja, qualquer tipo de comunicação aproveita tais esforços. Por isso, há a defesa direta a um direito a segurança e privacidade, que define seus termos sem limites de "objetos". Isto é que essas funções no mundo político "real"pertencem em geral como função reguladora do estado que tem monopólio da segurança e definem os limites da privacidade. Por exemplo, no caso da criptografia, e das ferramentas liberadas por comunidades como as dos sistemas BSD, as possibilidades de segurança e privacidade são disponibilizadas por atores independentes que ganham sua legitimidade por ser comunidades "livres". Assim, o poder polí-

default install!". Em 2007 o slogan era: "Only two remote holes in the default install, in a heck of a long time!". Para ter uma um ponto de comparaçao, tem atualizaçoes de segurança cada mês em sistema como Linux ou Windows.

tico tradicional somente pode interagir com elas por meio de restrições e nao de disponibilização ou monopólio.

3.4.3 Lógicas de inversão

A licença BSD, por favorecer uma colaboração sem restrições incluindo a apropriação do seu código para projetos proprietários não entra nas lógicas de inversão como podem ser achadas no projeto GNU-GPL.

3.4.4 Lógicas de colaboração

As comunidade BSD não estão explicitamente aberta como as do Samba ou de gNS. Quando se entra no site do projeto FreeBSD não tem um link ou um anúncio explícito chamando os usuários a contribuir ou participar. Os imperativos técnicos próprios a essas comunidades exigem um alto grau de competência comum apontada como 'elitista' pelas outras comunidades livres. Os atuantes consideram que a exigência técnica atrás do projeto os proíbe aceitar a ajuda de amadores cujo trabalho aparece como ma-acabado e provavelmente fraco em frente a suas exigências de estabilidade.

Esse 'profissionalismo' atrai mais ainda os investidores privados por propor condições de apropriabilidade muito vantajosa para as empresas desenvolvendo produtos fechados. A licença BSD permite o uso de um código que ela regula sem constringir nenhuma obrigação de retorno para as comunidades. Muitos exemplos de colaboração assim determinada resultaram produtos famosos ao mercado das novas tecnologias, dentro das quais Xbox, console de jogo desenvolvida pela Microsoft a partir de um kernel FreeBSD, PSP, console de jogo portável desenvolvida pela Sony a partir de um kernel NetBSD, MacOSX, sistema operacional da Apple desenvolvida a partir do sistema FreeBSD, entre outros.

Ainda, códigos BSD são usados para projetos livres, e incorporado ao sistema Linux, por exemplo. Isso coloca as comunidades BSD como plataforma de colaboração possível entre as lógicas, isto é, as mais proprietárias como as mais livres, e os usuários-desenvolvedores gostam de sublinhar que tal característica privilegia antes de tudo as características técnicas do sistema e permite um ambiente diversificado se manter. Nesse sentido, um comentário de um desenvolvedor no Forum da comunidade FreeBSD, nos informa: "É a hora de entender que FOSS é versátil. Você não queria que Linux fosse BSD, BSD fosse MacOSX, etc. Assim sendo, podemos ter a escolha e, por

definição, algo "bom para tudo" não pode ser bom."[18]. Desta forma, ilustramos uma lógica diferente na interpretação do que é o software livre e o que ele permite: A versatilidade de um meio-ambiente cuja diversidade técnica e legal permite a soluções diferentes de adaptarem-se adequadamente a necessidades diversas.

3.5 Comparando e analisando as características das pragmáticas

Ao comparar as pragmáticas da programação expostas aqui (Figura 3.1, 77 podemos observar que essas três comunidades interpretam diferentemente a sua participação ao movimento do software livre e que essas diferenças exprimam-se em primeiro lugar nas escolhas feitas na hora de codificar o software objeto das comunidades.

Essa diversidade técnica sublinha para seus atuantes uma representação diferente do que significa um código livre e aberto, o que motivam eles a atuarem nos projetos e, por fim, qual idéia eles tem do ambiente tecnológico maior no qual o SL/CA interage. Tais características constituem a personalidade social e política de cada comunidade e a diferença das outras. As escolhas realizadas formam um arcabouço político de uma comunidade, de seu código até sua atuação no campo das tecnologias. Que seja por volta de uma preocupação estética ou uma responsabilidade a respeito de um potencial regulador, os mesmos padrões de decisões estão colocados no processo comunitário de criação e promoção do software próprio a cada comunidade.

Além disso, alguns padrões estão aparecem com a leitura do quadro recapitulativo (Figura 3.1). Primeiramente, as lógicas de transgressões e de inversão parecem se excluir uma a outra. No caso das pragmáticas de liberdade, as lógicas de transgressões estão pouco recomendadas dentro do discurso da comunidade por ser reconhecidas como uma forma de incluir 'meio-termos' a que quer se constituir como uma alternativa comprometida a uma ideologia. Enquanto isso, as pragmáticas de abertura e de segurança desenvolvem bastante as características transgressivas dos seus modelos de desenvolvimento por permitir a inclusão do objeto transgredido no software codificado. Isto

[18]"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.".

Figura 3.1: Quadro recapitulativo

	Pragmática de liberdade (*gNewSense*)	Pragmática de abertura (*Samba*)	Pragmática de segurança (*BSD*)
Lógicas de transgressãço	0	Engenharia reversa	Procura de falhas
Lógicas de civismo	Alternativa inteira e autônoma às propostas não-livres	Interoperabilidade de tecnologias híbridas	Segurança-estabilidade do dispositivo
Lógicas de inversãço	Espaço jurídico próprio protegido pela mesma legitimidade legal que o domínio proprietário	0	0
Lógicas de colaboração	Comunidade aberta; Colaboração absoluta em Espaço hermético	Comunidade aberta; Colaboração sem limites próprios ao SL/CA	Versatilidade; Comunidades relativamente fechadas; Exploração comercial unilateral

é, no caso de Samba, a comunicação com os protocolos proprietários, e no caso de BSD, a defesa contra as falhas de segurança.

Por parte essas características das lógicas de transgressões determinam as das lógicas de civismo. Enquanto a ausência de transgressão na pragmática de liberdade produz a necessidade de construção de uma alternativa entendida como absoluta, a presença de tais lógicas nas pragmáticas de abertura e de segurança sublinham a construção de um desempenho técnico mais pragmático. Isto é, no caso do samba, uma contribuição à interoperabilidade dos sistemas Unix com o mundo proprietário, e no caso de BSD, a distribuição de ferramentas seguras e estáveis para a atividade de cada um na rede.

A respeito das lógicas de inversão, podemos observar que elas caracterizam especificamente o domínio livre GNU-FSF. Isto é que as lógicas de

inversão próprias às pragmáticas de liberdade como se acham na comunidade gNewSense, estão o monopólio de um software que se quer exclusivo. Por ter procurado seus meios de defesa contra o proprietário nos mesmos campos que ele, isto é, as licenças autorais, o domínio GNU que o projeto gNS vem sistematizar, procura inverter as proposta de sistemas operacionais tanto proprietárias como híbridas.

Por fim, as lógicas de colaboração parecem oferecer um bom espelho da esfera política de cada uma das comunidades, por ser o seu resultado concreto a respeito da sua capacidade a interagir com o ambiente tecnológico de software nos seus arredores. Por isso, enquanto as pragmáticas de liberdade promovem uma lógica de colaboração utópica, isto é, absoluta, mas num espaço restrito a realizar qualquer meio-termo, as pragmáticas de abertura e segurança interagem bastante com seus ambientes respectivos. Porém essas interações se diferenciam bastante nas suas naturezas. As pragmáticas de abertura fazem da colaboração um objeto do software que desenvolvem. O objeto da colaboração é atraído ao esforço tecnológico livre para ele se tornar acrescentado dele. Diferentemente, as pragmáticas de segurança promovem uma colaboração muito forte com qualquer tipo de ator enquanto o 'profissionalismo' dos sistemas é mantido.

Observamos então que dentro do próprio movimento SL/CA interagem oposições radicais de visão política e social, em termo de meios desenvolvidos, propósitos atribuídos, contribuições realizadas e colaborações constituídas. Esse conjunto heterogêneo, porém vincula a mesma figura ao interagir com sua esfera politica maior, como movimento unido, ou pelo menos designado como tal. As suas asas reformistas, progressivas, ou revolucionárias são misturadas numa mesma entidade social recolhida pelo senso comum sob o nome genérico de SL/CA. Por isso, umas das hipóteses que possam ser emitida aqui é que o que é convencionalmente designado como movimento deveria talvez ser estudado sob as propriedades de um público. Nesse sentido, vale mencionar o esforço, talvez contraditório, das comunidades SL/CA concorrentes e opostas sobre vários assuntos, como é o caso dos domínios designados como 'livre' ou open-source, para no trato do discurso manter uma identidade comum como público diversificado, cujas interações estruturam o panorama tecnológico dos softwares. Assim testemunha o fundador do projeto gNewSense, Paul O'Malley, numa entrevista pessoal:

> Os desenvolvedores de software em vários campos, isto é desenvolvedores do Linux, como de BSD, usam de licenças que res-

> peitam as quatro licenças. Ha muitos outros campos, mas esses dois estão suficientes para os propósitos dessa analogia. Cada um desses campos permite facilmente a cada um participar, e além de participar, de interpolar, num sentido próprio as ciências sociais, justificando às pessoas que se juntam a um campo particular que estão no campo certo graças as filosofias internas consistentes. Eles são muito similares, e somente em casos periféricos se diferenciam. Ambos declaram ter a validade absoluta, porem em função de suas diferenças, são entidades muito distintas.
>
> Eles são como gêmeos partilhando muito ADN, portanto que são entidades do software livre legítimos embora separados. Uma prova positiva disto é quando você segue o caminho de cada um, uma comunidade tem seu trabalho colocado em MacOSX e a outra no projeto Debian, e talvez em gNS[19]

Assim, apresenta-se ao entendimento do observador não um movimento político homogêneo, mas um público diversificado cuja dinâmica constitua-se a partir de um conjunto de movimentos concorrentes, e às vezes, até opostos. As filosofias desenvolvidas por cada campo do SL/CA são construções respectivas das projeções e representação que as várias comunidades produzem a partir das escolhas que eles realizam ao codificar seu software. Porém, um conjunto comum (o "ADN") contribuiu a promover um modo de desenvolvimento mais abstrato que é partilhado.

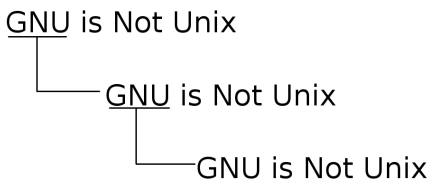
Todavia, qual seja híbrido ou a pretensão absoluta, uma noção das liberdades as quais devem responder uma licença são entendidas como colocadas ("as quatro liberdades"). Embora a comunidade seja "elitista"ou "educativa"em frente aos usuários-desenvolvedores, a uma mesma idéia desses pa-

[19]"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."

péis, como contributivo a um esforço de desenvolvimento livre e aberto. Por fim, portanto que as aplicações feitas desses softwares possam ser radicalmente diferentes, permanecendo livre ou tornando-se proprietárias, o conjunto primário de software fica acessível nos mesmos termos para qualquer um. Esses pontos comuns constituem um público, cujas modalidades comunitárias produzem os movimentos que fazem a dinâmica política do SL/CA.

Conclusão

Contaram-me uma piada que o Richard Stallman tem costume de fazer durante suas numerosas aparições públicas em congresso e seminários para promover o Software Livre. Ele vai até o quadro ou pega uma folha de papel e começa desenhar:



"O movimento do software livre é tão experto que até o nome dele é recursivo!", exclame-se ele no meio das risadas dos engenheiros, simpatizantes e amadores do SL/CA.

Há um pensamento contemporâneo que procura conceber a técnica como uma simples ferramenta. Assim, o contabilista, o maquinista, de maneira geral, o *técnico* possui uma prática hermética que serve uma a outra, ou seja, a do *pensamento* político, filosófico, social e econômico. Afirmamos as origens de tais ideais na filosofia antiga grega e sua concepção clássica do saber como um *dentro* que deve ficar independente de seu *fora*, a technê. A técnica, por sua vez, não participa da procura da felicidade, da beleza, do justo ou da verdade, mas a serve.

Urge então a imagem platônica do sofista, que usa o pensamento como uma técnica para tornar qualquer questão indiscutível, frequentemente a fins imorais. Numerosas historiografias desta época, porém, mostraram que a corrente sofista era desacreditada dos padrões da história das idéias e que se tratava na verdade de uma filosofia própria, pragmática e racional, privilegiando a análise das situações, dos lugares, dos eventos e das linguagens

de maneira concreta e não como um fim em si. Contudo, acham-se tais desenvolvimentos pejorativos em certas obras contemporâneas, como a do Bernard Stiegler, antigo aluna de Jacques Derrida e diretor do instituto de pesquisa e inovação (IRI) do centro Georges Pompidou, na França. Segundo ele, todo pensamento da técnica necessariamente ultrapassa os limites da filosofia. Para ele, a técnica não é nada.

Por tanto, se não é uma decisão política que conduz o movimento técnico científico atual e que recusamos o preconceito de imoralidade ou de inutilidade de uma technê autônoma, o que conduz então o movimento das redes e dos materiais que constrói nosso mundo já dependente dele? Acham-se elementos de resposta na obra do antropólogo Christopher Kelty e sua antropologia do geek através do estudo das significações culturais do movimento SL/CA. Segundo ele, a participação colaborativa e aberta de numerosos indivíduos permitiu a um público *recursivo* legítimo de se constituir. Como a função recursiva fatorial, função matemática (figura 3.2) que multiplica um número pelos inteiros que o precedem, e assim produz um novo numero maior; o público identificado por Kelty realiza seu presente invocando seus elementos já existentes.

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

O software é o conceito que as comunidades livres vêem invocar recursivamente. O software invoca o software, que invoca o software, que invoca o software. E assim produz programas, formatos, protocolos novos que participem do movimento tecnológico contemporâneo.

Poderíamos descrever tal movimento como uma experiência vivida do pensamento pragmático segundo o qual, "o esforço não é de praticar a inteligência, mas de intelectualizar a prática" (ELDRIDGE, 1998, p.5). Trata-se de uma recusa da procura para a certidão com categorias pré-conceituais para o contexto de análise presente. Num nível mais epistemológico, isto é, uma posição anti-retificação para a qual o conceito e a teoria não são objetos independentes, mas abstrações, cujo produto volta a experiência.

Nesse sentido, as primeiras linhas do manifesto Hacker dizem: "todas as classes temem essa abstração implacável do mundo, em cima da qual as fortunas ainda dependem". Entretanto, a "classe hacker"posiciona diferente, como notamos em seu discurso: "Somos os hackers da abstração. Nós produzimos novos conceitos, percepções e sensações hackeadas dos dados brutos".

Aqui, numa metáfora marxista de um mundo dividido por classes, a "classe Hacker" diferencia-se por uma relação inversa à abstração, como se ela não subisse sua cognição, mas a produziria a partir de um tratamento consciente das coisas num estado bruto.

Como observamos no capítulo primeiro desse estudo, as alternativas oferecidas pelo software livre sempre se constituíram pelo tratamento recursivo de uma matéria prima, Unix primeiro, depois os projetos GNU, BSD, Mozilla, para constituir um conjunto heterogêneo em termos de tecnologia, de regulamentos, como da esfera política e social.

O foco dado à figura do usuário-desenvolvedor no segundo capítulo nos permitiu achar os mesmo termos dentro da ética e as significações culturais do hacking. Nesse sentido, achamos exemplos nos trabalhos da antropóloga Gabriella Coleman, que nos mostraram como o agnosticismo político das comunidades SL/CA permitiu aos indivíduos de reinterpretar sua liberdade técnica e ética num contexto digital. A evolução, o ato, o movimento não se operam segundo idéias, preceitos, propósitos políticos preestabelecidos, mas pela prática de um presente. Este presente realiza-se num ato, o de programar, cujas pragmáticas pertencem tanto a uma responsabilidade legal, de regulamentação do espaço digital, como a uma responsabilidade artística, de manter e promover uma estética forte e elegante.

A relação à informação que nasce dessa pragmática da programação estrutura as especificidades das comunidades que tentamos apontar no capítulo três. Assim, as pragmáticas do ato de programar e as lógicas culturais da cultura do programador interagem para sublinhar pontos característicos das comunidades gNewSense, Samba e BSD, nas prioridades que dêem no processo de tratamento da informação. Que sejam as preocupações de liberdade, abertura ou segurança que confirmem o ato permanece recursivo, por tratar sempre as problemáticas comunitárias a partir do desenvolvimento das tecnologias e informações presentes e passadas.

Assim, constitui-se um público cujas modalidades dêem seus conteúdos políticos aos seus movimentos que se diferenciam e, às vezes, se opõem. O que permanece comum a esse público é seu tratamento recursivo das tecnologias para produzir novas tecnologias enriquecidas e modificadas. É esse movimento tecnológico que baseia o imaginário político de seus atuantes. A técnica é entendida como uma infra-estrutura, um conjunto de meios socialmente identificados e diferenciados para constituir sistemas operacionais, protocolos ou formato de arquivos que mantém um ambiente tecnológico, então dependente de várias éticas cujos atuantes se sentem responsável.

Portanto, por mais diversificada que seja, a ética do SL/CA permanece uma alternativa a outro modelo, proprietário, por expor os laços entre moral e técnica, infra-estrutura e superestrutura, sistema operacional e sistema social, como tantos objetos devendo ser construído por um público que ganha sua legitimidade sendo aberto e livre.

***

BIBLIOGRAFIA _

{Todos os endereços eletrônicos presentes nessa bibliografia foram accessados uma vez no dia 8 de julho de 2009.}

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**. Tese de doutorado em filosofia, Universidade de Pennsylvania, 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** (Cap.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**, (Cap. 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**. Monografia, 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>.