UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
FACULDADE DE CIÊNCIAS ECONÔMICAS
DEPARTAMENTO DE ECONOMIA E RELAÇÕES INTERNACIONAIS

JÚNIOR GOERGEN

SOFTWARE LIVRE COMO MOVIMENTO POLÍTICO DE INOVAÇÃO SÓCIO-ECONÔMICA: ALCANCES E LIMITES DE UMA EXPERIÊNCIA DE APROPRIAÇÃO COLETIVA DO TRABALHO COLETIVO

Porto Alegre
2015

JÚNIOR GOERGEN

SOFTWARE LIVRE COMO MOVIMENTO POLÍTICO DE INOVAÇÃO SÓCIO-ECONÔMICA: ALCANCES E LIMITES DE UMA EXPERIÊNCIA DE APROPRIAÇÃO COLETIVA DO TRABALHO COLETIVO

Trabalho de conclusão submetido ao Curso de Graduação em Ciências Econômicas da Faculdade de Ciências Econômicas da UFRGS, como requisito parcial para obtenção do título Bacharel em Economia.

Orientador: Profª. Dra. Gláucia Angélica Campregher

Porto Alegre

2015

Ficha catalográfica

CIP - Catalogação na Publicação

Goergen, Júnior
SOFTWARE LIVRE COMO MOVIMENTO POLÍTICO DE INOVAÇÃO SÓCIO-ECONÔMICA: ALCANCES E LIMITES DE UMA EXPERIÊNCIA DE APROPRIAÇÃO COLETIVA DO TRABALHO COLETIVO / Júnior Goergen. -- 2015.
63 f.

Orientadora: Gláucia Angélica Campregher.

Trabalho de conclusão de curso (Graduação) -- Universidade Federal do Rio Grande do Sul, Faculdade de Ciências Econômicas, Curso de Ciências Econômicas, Porto Alegre, BR-RS, 2015.

1. Software livre. 2. Open source. 3. Modelo de negócios. 4. Cultura hacker. 5. Inovação. I. Angélica Campregher, Gláucia , orient. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da UFRGS com os dados fornecidos pelo(a) autor(a).

JÚNIOR GOERGEN

SOFTWARE LIVRE COMO MOVIMENTO POLÍTICO DE INOVAÇÃO SÓCIO-ECONÔMICA: ALCANCES E LIMITES DE UMA EXPERIÊNCIA DE APROPRIAÇÃO COLETIVA DO TRABALHO COLETIVO

Trabalho de conclusão submetido ao Curso de Graduação em Economia da Faculdade de Ciências Econômicas da UFRGS, como requisito parcial para obtenção do título Bacharel em Economia.

Aprovada em: Porto Alegre, 23 de junho de 2015.

BANCA EXAMINADORA:

Profa. Dra.
Gláucia Angélica Campregher – Orientadora
UFRGS

Profa. Dra. Ana Lúcia Tatsch
UFRGS

Prof. Dr. Fabian Scholze Domingues
UFRGS

AGRADECIMENTOS

Primeiramente, meu agradecimento é para a minha para minha mãe, Teresinha da Silveira, que me deu todo o suporte, apoio e incentivos necessários para que eu conseguisse concluir mais esse ciclo da minha vida.

Gostaria, também, de fazer um agradecimento à professora Gláucia Angélica Campregher, por ter aceitado orientar esse trabalho sobre um tema ainda pouco difundido entre economistas. Seus conselhos e ideias de como conduzir o trabalho permitiram que eu tivesse a clareza e a confiança de que estava no cominho certo.

Aos colegas de trabalho e coordenadores da Associação Software Livre.Org pela ajuda nesse momento. Aos amigos Daniel Mossatte Goulart e Pablo Diogo Rex Cardoso pelas palavras de incentivo e camaradagem. E a minha querida namorada Thais Della Colletta de Aguiar, a qual foi uma grande companheira nesse etapa da minha vida. Sempre me incentivando para que produzisse o meu melhor com paciência e generosidade, mesmo nas ocasiões em que abriu mão de momentos de lazer e descanso.

Mas sou, e talvez serei sempre, o da mansarda,
Ainda que não more nela;
Serei sempre o que não nasceu para isso;
Serei sempre só o que tinha qualidades;
Serei sempre o que esperou que lhe abrissem a porta ao pé de uma parede sem porta
E cantou a cantiga do Infinito numa capoeira,
E ouviu a voz de Deus num poço tapado.
Crer em mim? Não, nem em nada.
Derrame-me a Natureza sobre a cabeça ardente
O seu sol, a sua chuva, o vento que me acha o cabelo,
E o resto que venha se vier, ou tiver que vir, ou não venha.
Escravos cardíacos das estrelas,
Conquistámos todo o mundo antes de nos levantar da cama;
Mas acordámos e ele é opaco,
Levantámo-nos e ele é alheio,
Saímos de casa e ele é a terra inteira,
Mais o sistema solar e a Via Láctea e o Indefinido.
(Trecho do Poema Tabacaria de Ávaro de Campos – heterônimo de Fernando Pessoa)

RESUMO

Este trabalho se propõe a investigar os alcances e limites do *Software* Livre e da cultura do *copyleft*. Através de uma investigação das origens da indústria do *software*, busca-se compreender como o *software* se tornou uma mercadoria e como a consolidação dessa indústria provocou uma reação entre aqueles que queriam e podiam fazer do *software* um produto de circulação e co-produção livres, que foi a fundação de uma agremiação política - o Movimento Software Livre. Ocorre que a produção coletiva e colaborativa com o tempo se mostrou mais eficiente e menos custosa chamando a atenção das empresas tradicionais e suas práticas de *copyright* que passam a trabalhar com tecnologias *Open Source* e provocam a divisão do Movimento Software Livre em dois grupos: grupo *free* e grupo *open*. A questão do que representou esta ruptura do ponto de vista econômico e político-ideológico é o foco do trabalho. Por fim, será analisado se e como a cultura do *copyleft* pode conviver com a indústria do *software* através de uma análise do perfil e das motivações dos participantes dos projetos de *software* livre e de código aberto e também através dos modelos de negócios adotados pelas empresas que prestam serviços em *software* livre/aberto.

Palavras-chave: Software livre. Open source. Modelo de negócios. Cultura hacker. Inovação.

ABSTRACT

This study aims to investigate the scope and limits of free software and copyleft culture. Through an investigation of the software industry origins, we seek to understand how the software has become a commodity and as the consolidation of this industry provoked a reaction among those who wanted and could make the software a product of free movement and co-production, which was the foundation of a political group - the Free Software Movement. It turns out that the collective and collaborative production over time is more efficient and less costly calling the attention of traditional companies and their copyright practices that go on to work with Open Source technologies and provoke the Free Software Movement division into two groups: free group and open group. The question of what accounted for this rupture from the point of economic and political-ideological view is the focus of the work. Finally, it will be examined whether and how the copyleft culture can live with the software industry through a profile analysis and motivations of participants in free software projects and open source and also through business models adopted by companies provide services in free / open source software.

Keywords: Free Software. Open source. Business model. Hacker culture. Innovation.

SUMÁRIO

1. INTRODUÇÃO

O presente trabalho se insere dentro de um debate que acontece na sociedade e na academia e dentro desta envolve diversas ciências incluindo as sociais, trata-se do debate em torno do fenômeno do Software Livre. Este longe de ser um desenvolvimento tecnológico ligado às chamadas tecnologias da informação, ou um produto das indústrias desta, tem uma origem social e uma prática, de certo modo, de confronto com o Capital. Deve-se a isto ser o *Software* Livre um *movimento tecnosocial*.

Herdeiro da ética *hacker*, o *Software* Livre (SL) é responsável por uma das maiores inovações dentro da tecnologia, mas também dentro dos movimentos sociais, na atualidade – qual seja a viabilização do compartilhamento da informação e do conhecimento. Essa viabilização se dá através da disponibilização de licenças de uso e códigos de programação que estão por trás de toda uma cultura que viceja na internet e que, se começa pelo compartilhamento de arquivos, alcança já o de bens e serviços.

De fato, os defensores e desenvolvedores de SL fundaram um movimento político Movimento Software Livre (MSL) com a mesma finalidade. Seus fundadores desenvolveram uma filosofia de base que carrega algo de subversivo: a cultura do *copyleft* como antítese do *copyright*, uma vez que este é o reconhecimento do direito autoral fundamentando uma propriedade, concedendo ao autor direitos de exploração comercial daquilo que ele desenvolveu em termos de conhecimento objectivado, e que aqui é objeto de socialização. Basta lembrar que é através do MSL que surgem outros tipos de licenças livres para a propriedade intelectual e que se promovem diversos fóruns de discussão de experiências

De fato, a ideia de compartilhar, cooperar e desenvolver ao máximo a liberdade de criar e trocar conhecimento, é tão antiga quantos são os computadores. No inicio da informática, era bastante comum o compartilhamento e a cooperação solidária entre os desenvolvedores. Dito em outras palavras, o *software* não possuía valor de troca, apenas valor de uso. Essa realidade se transforma no momento em que o *software* passa a ser vislumbrado como uma mercadoria, ou seja, o valor de troca aparece predominantemente. Foi justo este o cenário de surgimento do Movimento S*oftware* Livre. Pois, o objetivo maior deste trabalho é responder à pergunta de como se deu esta predominância que, adianta-se, não ocorreu de uma vez e nem sem um enfrentamento com aqueles que não apenas se julgavam representantes dos valores éticos difundidos no momento anterior, como eram (e são) parte preponderante da mão de obra dos desenvolvedores

das tecnologias em jogo.

Mas não se trata de contar aqui uma história finda. Ainda hoje se trava uma luta entre a *cultura do copyleft* e a lógica da valorização capitalista da apropriação privada do trabalho coletivo sustentada, entre outras, pela lei do *copyright*. O que significa que importa explicitar neste trabalho que cultura é aquela e que leis são estas. Importa questionar também se estas além de se contraporem não vão, como desdobramento dos embates e por dificuldades próprias a cada uma, passando a conviver e até interagir. O que leva a uma pergunta outra que é se com isto o MSL ganha ou perde em radicalidade *vis a vis* as suas propostas iniciais.

Dito isso, esta monografia se divide em três capítulos. O Capítulo 1 trata dos primórdios da criação dos computadores como máquinas cujas operações são recorrentemente programadas, que se comunica por isso com programadores e mesmo com outras máquinas. A história da "indústria do *software*" desde os anos 1940 irá mostrar como nesta, mais que em qualquer outra indústria, a genialidade de alguns poucos pioneiros é menos responsável por seu rápido crescimento e evolução que o trabalho coletivo dirigido por aqueles (industriais no sentido capitalista) ou livremente articulado.

No segundo capítulo, apresentam-se as bases filosóficas do MSL, as técnicas de produção que defendem e os desdobramentos que acarretam na economia. Destaca-se a invenção da *General Public Licence* (GPL) e, por decorrência, da cultura do *copyleft*. Mostra-se como os defensores de códigos abertos, principalmente as figuras midiáticas de Richard Stalman e Linus Torvalds, junto com todo um universo de programadores, conseguiram desenvolver um *software* de alta complexidade construído colaborativamente e com uma licença que coletiviza sua propriedade intelectual. Vê-se ainda como a indústria tradicional reage seja se referindo a este como um "câncer" ou um "vírus" [1]. E, por fim, aponta-se para as contradições internas vividas recentemente pelo movimento.

No Capítulo 3, e último, tem-se a pretensão de fazer uma análise de até que ponto o MSL tem se viabilizado em suas propostas originais ou tem se conformado à lógica de mercado. A investigação inicia traçando um paralelo entre os modelos de negócios da tradicional indústria da TIC e do *Software* Livre/*Open Source* procurando exemplificar as diferenças e semelhanças. A

[1]Em 2001, Steve Ballmer, concedeu uma entrevista ao jornal Sun Times de Chicago (USA) e ao ser perguntado pelo repórter se ele enxergava o Linux e o Open Source como uma ameaça ele respondeu que considerava o Linux um câncer por que "contamina" tudo o que toca. Ballmer se referia a licença que obriga todos os trabalhos derivados do código-fonte original também serem abertos. Steve Ballmer foi presidente da Microsoft de 1998 até 2014.

tentativa de transformar o *software* livre/aberto em uma mercadoria ao mesmo tempo em que os projetos se esforçam para continuarem autônomos e produzindo de forma descentralizada, livre e cooperativa. Porém, para compreendermos adequadamente os projetos precisamos, antes, entender como funcionam as motivações de seus membros. Para isso, utilizaremos duas pesquisas realizadas dentro das comunidades diretamente com seus desenvolvedores. Uma delas está circunscrita ao Brasil e outra abrange Estados Unidos, Canadá e Europa Ocidental. Através dessas pesquisas de campo, conseguiremos compreender o que justifica o trabalho voluntário, a criação de bens de alta complexidade fora das estruturas empresariais e sem pretensão de transformá-la em mercadoria. Contudo, também poderemos compreender como as empresas de TIC se relacionam com os membros das comunidades garantindo patrocínios a determinados projetos que possuem um valor comercial imediato. Questiona-se nesse capítulo os modelos híbridos (liberdade de código-fonte, busca por eficiência e superioridade técnica, desejo de sucesso comercial) como dando espaço para o trabalho colaborativo apenas na medida em que este possa ser apropriado privadamente. Por último, faremos uma investigação da produção fora do mercado, de como o *software* livre e o modelo de desenvolvimento bazar está distante de ser domando pelo capital e apontam caminhos para novas relações de trabalho, novas formas de produção, novos modelos de negócios e, tudo isso, com a propriedade livre.

Ao final do trabalho, apresenta-se uma Conclusão enfatizando os principais elementos abordados.

2. O NASCIMENTO DA INDÚSTRIA DO SOFTWARE E A GÊNESE DO PROJETO GNU

A informática nasceu, como o estudo de um ramo do saber humano, no início do século XX e no decurso de seu fértil progresso a humanidade observou grandes e distintas fases de invenções que mudariam, para sempre, a forma de comunicação e de organização da informação que foram o computador e a internet. Entretanto, as invenções, separadas cronologicamente no tempo estavam interligadas por uma estreita afinidade, embora não existisse uma continuidade perfeita. Em seu livro, História da Informática, Philippe Breton (1991), promove uma diferenciação da história da informática em três períodos, a qual adotaremos como referência no decorrer desse trabalho, sendo a primeira informática, iniciando-se na década de quarenta e indo até o início dos anos sessenta, foi caracterizada pelo financiamento e pela introdução dos princípios essenciais para a invenção do computador; a segunda informática, iniciando-se na década de sessenta e se estendendo até a década de setenta, caracterizou-se pelo estabelecimento dos grandes sistemas centralizados; a terceira informática, a que conhecemos atualmente, é a diversificação dos meios e dos métodos, das redes e da microinformática, dos pequenos e grandes sistemas.

2.1 OS PRIMEIROS COMPUTADORES E O PRINCÍPIO DA INFORMÁTICA

Na primeira informática é onde surgem os primeiros computadores, os quais são o resultado de uma extensa cadeia evolutiva de saberes que se acelerou durante o século XX. Os computadores eletrônicos nascem durante Segunda Guerra Mundial através do esforço de guerra empreendido pelos Estados Unidos e pela Inglaterra para vencer a Alemanha Nazista. A guerra foi um importante catalisador para desenvolvimento dos primeiros computadores modernos. A aplicação tecnológica na guerra não era nenhuma novidade, porém os volumosos investimentos dedicados à pesquisa científica abreviaram o tempo de maturação que as descobertas e invenções teriam em tempos de paz. Como Hobsbawm (2006) descreve a seguir, a Segunda Grande Gerra acelerou os avanços científicos, pois a guerra “[...] adiantou visivelmente a tecnologia, pois o conflito entre beligerantes avançados era não apenas de exércitos, mas de tecnologias em competição para fornecer-lhes armas eficazes e outros serviços essenciais [...]”. Hobsbawn (2006) conclui que muitas invenções de guerra tiveram aplicações mais imediatas na paz como os

computadores, por exemplo.

A Segunda Grande Guerra e também a guerra fria representaram a causa decisiva que possibilitou a invenção do computador moderno e o surgimento da internet. Se estudarmos detalhadamente a história da informática[2] verificar-se-á a intimidade que os projetos de pesquisa científica que deram origem aos computadores, à inteligência artificial e às de redes, executados dentro das universidades, foram forjados com financiamento militar. Importantes e diferentes cientistas se dedicaram à busca pela construção de autômatos - "cérebros artificiais" - que reproduzissem o funcionamento do cérebro humano (BRETON, 1991). A partir de 1945, avançou-se na tarefa de construir potentes artefatos que funcionam como um automatismo de programação que irá permitir ao mesmo tempo efetuar cálculos aritméticos e processar informação de forma lógica. Os computadores, portanto, foram "[...] inicialmente concebidas como máquinas para realizar cálculos matemáticos com eficiência e rapidez, sua capacidade para ordenar informações diversas é que granjeou aos computadores seu lugar no cotidiano social" (MAGALHÃES, 1997).

Se a Segunda Grande Guerra funcionou como catalisador para acelerar o aparecimento do computador, a guerra fria desempenhou um papel semelhante para o surgimento de uma tecnologia que mudou extraordinariamente a comunicação, inaugurando a era da informação em larga escala, a internet. Durante o final dos anos 1950, foi criada a *Advanced Research Projects Agency* – ARPA – como uma resposta do governo dos Estados Unidos ao lançamento do satélite *Sputnik* pela URSS. Um dos ousados projetos assumidos pela ARPA estava projetar um sistema de comunicação invulnerável a ataque nuclear. Em 1969 é lançada a ARPANET, uma rede descentralizada, composta por vários outras redes. Seu funcionamento é explicado por Castels (1999) da seguinte maneira:

> Com base na tecnologia de comunicação por comutação de pacotes, o sistema tornou a rede independente de centros de comando e controle, de modo que as unidades de mensagens encontrariam suas rotas ao longo da rede, sendo remontadas com sentido coerente em qualquer ponto dela. (CASTELS, 1999, p. 375).

A partir de 1950, inicia-se a venda dos computadores para o mercado civil dos Estados Unidos,

[2] Os componentes de hardware evoluíam de forma muito mais rápida do que os softwares. Vale lembrar que as primeiras linguagens de programação reais começaram a surgir no final dos anos 1950 (BRETON, 1991). Essas linguagens são as chamadas linguagens de "alto nível", mais próximas da linguagem humana e do entendimento do programador do que da máquina. A escrita e a leitura de códigos-fonte e de linguagens de programação só começaram a se tornar difundidas em meados dos anos 1970.

para algumas poucas empresas privadas de grande porte, porém era o mercado estatal estadunidense o maior comprador e o maior financiador de pesquisas na área da computação, principalmente, as forças armadas (BRETON, 1991). Apesar da abertura de um mercado civil, nesse período, a informática era muito técnica e centralizada na máquina. Os computadores ainda estavam muito distantes de possuírem a aceitação popular dos dias atuais. Os modelos comercializados a partir de 1951 eram aparelhos grandes e barulhentos que ocupam uma sala inteira e funcionavam com cartões perfurados.

O início dos anos 1960 inaugura a "segunda fase informática". A década de 1960, portanto, representariam um momento de transição de uma informática absolutamente técnica que tinha como principal cliente as forças armadas para uma "informática dos especialistas", como o próprio Breton afirma. É o momento em que as "[...] máquinas, e os seus *softwares*, passam a ser pensadas como ferramentas para produzir um novo estilo de gestão e de organização empresarial, uma ferramenta para processamento da informação" (TORRES, 2013).

Durante esse período, a informática se torna um negócio estratégico, vai aumentando sua importância econômica e a participação na vida empresarial e social se constituindo ordenadora da sociedade. Ao venderem computadores, as empresas de tecnologia também vendiam a nova ideia relacionada a modernidade. É nesta época que, segundo Torres (2013), a imagem de máquina de calcular que o computador tinha, vai, aos poucos, sendo substituída por uma imagem de máquina de processar e ordenar a informação. Os franceses marcam essa mudança de status dos computadores através da preferência do uso da palavra *ordinateur* ao invés de *calculateur*. Dessa forma, Philippe Breton (1991) explica:

> Os computadores, desde então, não são mais encarados como simples máquinas de calcular ou para a confecção de folhas de pagamento, mas como máquinas de processar a informação. Os franceses, tão preocupados com o peso das palavras, começavam a adotar o termo "ordinateur" (proposto pelo professor Jacques Perret em 1955 a pedido da IBM francesa para substituir o inadequado "calculateur", do mesmo modo que computer substituirá calculator), que transformava essa máquina – sobretudo na expressão "ordinateur universel" - numa verdadeira máquina de colocar ordem, de organizar. A partir daí, a questão não era mais a máquina enquanto tal, mas o que ela permitia que fosse processado, a informação. (BRETON, 1991, p. 220).

O início da comercialização de computadores e o estabelecimento de uma indústria desencadearam dois processos relevantes que ocorrem no final dos anos 1960 e início dos anos 1970 que foram impulsionadores do desenvolvimento dessa indústria: primeiro, o registro da primeira patente de *software* num processo antitruste contra a IBM, que culminou na mudança de

seu modelo de negócios, baseado no *software* compartilhado e/ou gratuito, e teve grande influência sobre a formação de um modelo de negócios baseado no *software* de código proprietário; segundo a revolução da microinformática nos anos 1970 na Califórnia.

2.2 O NASCIMENTO DA INDÚSTRIA DO *SOFTWARE:* DA PRIMEIRA PATENTE DE *SOFTWARE* AO *UNBUNDLING* DA IBM

Até a metade dos anos de 1960 não havia uma indústria especializada em produzir *software* e outra especializada em produzir *hardware*. As máquinas que eram vendidas, na maioria das vezes, vinham com o *software* aberto, ou seja, não havia, por parte do fabricante, a preocupação (ou a intenção) de cobrar pelo uso e restringir o acesso dos usuários ao código-fonte[3] dos programas. Até por que o *software* não representava uma fonte de lucro para os fabricantes de computadores, dito de outra forma, o *software* não era compreendido como mercadoria. Um exemplo dessa conduta empresarial é a IBM que, ao vender um computador, distribuía gratuitamente seus *softwares*, com seus respectivos código-fonte sem exigir o pagamento pela licença de uso, inclusive alguns *softwares* eram colocados em domínio público. Uma explicação para esse fato é que o poder de processamento e o tamanho dos computadores avançavam mais rapidamente do que o desenvolvimento das linguagens de programação e da capacidade dos programadores em explorá-los (CAMPBELL-KELLY et al, 2014).

Além disso, não existiam, na legislação, impedimentos legais para que as empresas incorporassem aos seus *softwares* outros *softwares* produzidos por outras empresas. Existiam grupos de usuários que cooperavam uns com os outros trocando programas e dicas de programação. A IBM facilitava essa troca de programas entre os usuários mantendo uma biblioteca de programas desenvolvidos para os seus clientes. Estes programas raramente eram usados da forma como foram desenvolvidos originalmente, mas os programadores e usuários desses programas tinha liberdade para adaptar e melhorá-los (CAMPBELL-KELLY et al, 2014).

Entretanto, esses softwares não eram livres, pois esse conceito só seria elaborado por Richard Stallman na década de 1980. Esses *softwares* foram desenvolvidos para integrarem a

[3]O código-fonte é um conjunto de instruções escritas de forma ordenada, não compilada para o código de máquina, em uma determinada linguagem de programação.

mercadoria “computador” e desempenhar tarefas e não como uma mercadoria dentro de outra mercadoria. O fato é que ainda não havia sido vislumbrada a lucratividade da propriedade intelectual do modelo de negócios do software proprietário. Portanto, não se pode negar que havia uma cultura de trocas de softwares dentro do ambiente corporativo que utilizava computadores. Essas práticas evidenciam que essa cultura não era restrita aos ambientes de clubes *hackers* e/ou de apaixonados por tecnologia. A liberdade de acessar os códigos dos programas e modificá-los era muitas vezes explícita, não necessitando, portanto, de algo que a reafirmasse, como acontece hoje com o *software* livre (TORRES, 2013)

Apesar do compartilhamento de *software* ser relativamente rotineiro entre os usuários, algumas empresas de tecnologia não possuíam o mesmo entendimento e percebiam essa prática como prejudicial aos negócios à medida que a indústria voltada para a produção e venda de *softwares* foi se desenvolvendo, o modelo de *software* proprietário, atualmente predominante (onde seu código-fonte é protegido por *copyright* eliminando a possibilidade de adaptação ou de melhoria do *software* pelos seus usuários), foi se estabelecendo como um padrão comercial. E mesmo a própria IBM teve que se adaptar a essa nova realidade reestruturando a forma como produzia e distribuía seus *softwares*.

Para compreendermos, adequadamente, o processo em que *software* se transforma numa mercadoria há, pelo menos, três eventos que precisam ser compreendidos, porque foram importantes no desenvolvimento de uma indústria e de um modelo de negócios. O primeiro está relacionado à primeira patente de *software* registrada em 1968; e o segundo com a decisão da IBM de separar a venda do *hardware* dos serviços de *software*, também em 1968; e o terceiro com a carta aos *hobbistas*, na qual Bill Gates defende o uso de *copyright* como mecanismo de proteção da propriedade intelectual da linguagem de programação BASIC.

A IBM, que já era a maior empresa de tecnologia da época[4], obteve muito sucesso no mercado comercializando computadores contendo *softwares* gratuitos e compartilháveis. Após o lançamento do IBM System/360, que estabeleceu uma arquitetura padrão de mercado, algumas

[4]Philipe Breton (1991) comenta que nos anos 1970 a indústria informática dos Estados Unidos dominava o mercado mundial de computadores tendo à frente, como companhia que dominava essa indústria, a IBM. Recebeu por isso o apelido de “Big Blue”, possivelmente por causa da sua cor oficial, que é o azul; ou “Branca de Neve”, apelido que se referia aos “sete anões”, as setes grandes outras principais empresas de informática: Burroughs, UNIVAC, NCR, Control Data, Honeywell, General Electric e RCA. No final dos anos 1960, essas empresas juntas tinham pouco mais de 36% do mercado mundial, enquanto a IBM tinha, sozinha, 50%. Essa situação ainda pioraria no final dos anos 1970 e começo dos 1980, quando o percentual dessas empresas caiu para 11%.

empresas de software começaram a explorar a ideia de desenvolver seus programas e converter alguns de seus programas existentes para vender ao mercado de usuários de computadores, especialmente os usuários de computadores da IBM. Dentro dessa estratégia, a ADR foi fundada em Princeton em 1959, com o intuito de vender serviços de desenvolvimento de software para fabricantes de computadores. Em 1964, a ADR desenvolveu um *software* chamado Autoflow (um programa para fazer fluxogramas) que foi oferecido, num primeiro momento, aos executivos da RCA, que era uma empresa em ascensão, para que eles comprassem o software e distribuísse aos seus clientes. No entanto, a RCA não demonstrou interesse em adquirir o software por considerar o preço alto demais. Apesar de não ter a intenção de vender o *software* como produto, a ADR tentou vender o Autoflow diretamente aos usuários de computadores da RCA, que eram em torno de cem empresas, sob pena de não recuperar o investimento de cerca de 10 mil dólares feito no desenvolvimento do programa.

As vendas ficaram abaixo da expectativa, talvez, porque a maioria das pessoas não estava habituada a pagar por um *software*. No início de 1964, a ADR resolve voltar seus esforços em adaptar o Autoflow para o conjunto de usuários de computadores IBM cujo mercado era muito maior do que o de usuários da RCA (GOETZ, 2002). A ADR conseguiu vender muito bem considerando que em 1965 não era comum uma empresa vender cópias de um programa de computador. Entretanto, esse mercado com milhares de clientes em potencial foi severamente limitado pela crença por parte desses clientes que esse *software* que estava sendo vendido a US $2.400,00 poderia ser obtido gratuitamente na IBM (JOHNSON, 1998).

A IBM já havia desenvolvido um programa similar ao Autoflow, batizado de Flowcharter, o qual não rodava automaticamente, ou seja, era necessário algum trabalho do usuário para que o programa funcionasse. Os usuários esperavam que a IBM fornecesse o Autoflow gratuitamente ou implementasse suas funcionalidades no Flowcharter que já era uma solução IBM, assim, além de não ter a necessidade de comprar o programa desenvolvido pela ADR, a IBM, atendendo aos pedidos de seus clientes para aprimorar o Flowcharter retirava, efetivamente, a ADR do mercado. Martin Goetz, sócio-fundador da ADR, temendo essa possibilidade, registrou a primeira patente sobre *software* da história. A patente sobre o Autoflow foi solicitada em 1965 e foi confirmada em 1968. A IBM assim ficaria impedida de produzir um programa igual ao da ADR, pois violaria a patente registrada pelos proprietários Autoflow (TORRES, 2013).

Seus concorrentes enxergavam a IBM como uma grande propagadora do modelo de negócios de software gratuito e compartilhado e queriam acabar com essa prática. O IBM's

Unbundling (ação promovida pela IBM para cobrar separadamente o *hardware* e os serviços de *software*), como foi chamado esse acontecimento, mudou significativamente a história da indústria do *software*. Martin Goetz no artigo *Memoirs of a software Pioneer* (Memórias de um Pioneiro do *Software*) comenta os antecedentes da tomada de decisão da IBM:

> Lembro-me de muitas reuniões em 1967 e 1968 com os advogados do Departamento de Justiça que estavam preocupados com o domínio da IBM no campo do hardware. Durante 1968, o Departamento de Justiça entrevistou um número de empresas independentes de software e em Janeiro de 1969 entrou com uma ação contra a IBM. A denúncia, que abrangeu o domínio da IBM no hardware, também alegou que a IBM tinha impedido o crescimento da indústria de software como resultado da agregação de hardware e software, a partir dos anos 1960. […] Em abril de 1969, ADR processou a IBM para monopolizar a indústria de software. Nós estávamos tentando proteger o nosso futuro e, ao mesmo tempo, obter indenização pela redução das receitas no mercado programa de computadores. Apenas dois curtos meses depois, em junho de 1969 - seis meses após o ajuizamento da ação do Departamento de Justiça - IBM anunciou que, a partir de Janeiro de 1970, iria separar seus sistemas e aplicações de *software* [...]. (GOETZ, 2002, p. 52-53, tradução nossa[5]).

Não há dúvidas que o u*nbundling* da IBM ajudou a legitimar a conversão do *software* numa mercadoria e sua comercialização como se fosse um bem tangível e, portanto, rival. Também foi importantíssimo para o desenvolvimento do modelo de negócios que o *software* proprietário adota atualmente, principalmente porque teve como uma de suas principais implicações a necessidade de proteger os softwares de usos não autorizados pelos desenvolvedores, que pudessem causar prejuízos financeiros às empresas. Isso envolve cercear o acesso que as pessoas poderiam ter a ele ou mesmo ao seu modo de funcionar, através do estudo do seu código-fonte. Por conta disso, as empresas desenvolvedoras iniciaram um processo de licenciamento de seus programas, protegendo-os, através de um mecanismo jurídico de proteção da propriedade intelectual conhecido como *copyright*. Cada usuário possuiria uma licença de uso individual que o impediria de compartilhar o *software* com qualquer outra pessoa, muito menos revendê-lo a quem quer que fosse.

---

[5]No original: I remember many meetings in 1967 and 1968 with Justice Department attorneys who were concerned about IBM's dominance of the mainframe hardware field. During 1968, the Justice Department interviewed a number of independent software companies and in January 1969 brought suit against IBM. The complaint, which covered IBM's dominance of hardware, also alleged that IBM had hindered the software products industry's growth as a result of bundling the hardware and software, beginning in the 1960s. […] In April 1969, ADR sued IBM for monopolizing the software products industry.We were attempting to protect our future and, at the same time, collect damages for reduced revenues in the flowchart market during the 1960s.Just two short months later, in June 1969 — six months after the filing of the Justice Department suit—IBM announced that, beginning in January 1970, it would unbundle its systems and application software […].

Ao colocar esses dois acontecimentos, nos longínquos anos 60, como cruciais para o desenvolvimento comercial da indústria do software, procura-se estabelecer quais fatos contribuíram para o estabelecimento e consolidação desse modelo de negócios influente e poderoso comercialmente. Mas também não se pode atribuir apenas a eles a responsabilidade pelo que viria a se transformar numa indústria. Esses dois eventos não foram os únicos que concorreram para a mercantilização dos softwares. Durante os anos 1970, na Califórnia, dá-se início ao que Philippe Breton chamou de “revolução dentro da revolução”

2.3 A ÉTICA HACKER, A MICROINFORMÁTICA E A POPULARIZAÇÃO DA INFORMÁTICA

Desde os anos cinquenta o computador era compreendido como símbolo de grandes instituições centralizadas e burocráticas. Porém esse tipo de pensamento começou a mudar durante os anos sessenta. Conforma afirma John Markoff “a computação deixou de ser julgada como uma ferramenta de controle burocrático para ser adotada como símbolo de expressão individual e libertação” (MARKOFF, 2005). Durante os anos sessenta, os Estados Unidos experimentavam uma grande agitação política e social. Temas como direitos civis, direitos das mulheres, ecologia e os movimentos anti-guerra e psicodélicos contribuíram para o surgimento de uma contracultura que colocou em xeque alguns valores cristalizados da sociedade estadunidense.

As tecnologias da informação e comunicação que utilizamos atualmente devem sua forma a esse momento desobediente. Breton descreve esse momento como uma “[...] mistura bem saborosa de esquerdismo eventualmente marxista, de zen-budismo, de ecologia de 'sobrevivência', de rock e música eletrônica, de ficção científica permeada de volta às origens” (BRETON, 1991). Movimentos políticos como a *New Left* e o movimento de contracultura influenciaram tanto grupos que consideravam a tecnologia uma ferramenta que poderia levar ao progresso social quantos grupos que eram anti tecnologia com uma atuação que lembrava os luddistas, quais pregavam uma volta a terra. Um dos projetos mais populares foi o *Whole Earth Catalog*, publicado pela primeira vez em 1968, o qual influenciou toda uma geração a pensar o uso da tecnologia de maneira democrática e descentralizada. Fred Turner afirma que “[...] com o tempo, a rede de leitores e contribuidores que se desenvolveu juntamente com Catálogo ajudou a criar as condições culturais sob as quais os microcomputadores e as redes de computadores

poderiam ser imaginadas como ferramentas de libertação.” (TURNER, 2006:73)[6].

O impacto dessa mistura de ativismo político e tecnologia na região oeste dos Estados Unidos podem ser vistos claramente nas histórias pessoais de muitos desses pioneiros da indústria de computadores. Na verdade, as decisões pessoais frequentemente tiveram consequências históricas. Na Califórnia foi onde se concentraram empresas de microeletrônica que, através da compactação e miniaturização de semicondutores e chips, possibilitaram o surgimento de computadores compactos. Entretanto, a IBM poderia desenvolver um produto como esse, mas o microcomputador não correspondia, enquanto projeto ao entendimento que a companhia tinha da informática do futuro. Além disso, não estava nítido se haveria mercado para esse tipo de produto. Dessa maneira, o desenvolvimento e a popularização dos computadores pessoais deixou a costa leste – berço da computação dos Estados Unidos - e se deslocou para a Costa Oeste, precisamente, no lugar que ficou conhecido como Vale do Silício. Markoff (2005) comenta que “[...] o transistor foi inventado pela AT&T Bell Laboratories em New Jersey, mas a gigante de telecomunicações mais tarde foi forçada a licenciar a invenção livremente sob os termos de um acordo antitruste com o Departamento de Justiça. A própria existência do Vale foi possível graças a disponibilidade forçada do transistor.”

Influenciados por esse contexto, os *hackers* californianos enxergavam o computador como uma ferramenta de democratização do acesso à informação. Os computadores pessoais, que foram concebidos para pertenceram a um único indivíduo, surgiram da rejeição da autoridade e da crença de que o criativo espírito humano triunfaria sobre a rígida tecnologia das grandes empresas. (MARKOFF, 2005; BRETON, 1991). No entanto, além desse sentimento revolucionário, não podemos ser ingênuos e desconsiderar que a ideia empreendedora de se tornar um fabricante de microcomputadores e fazer fortuna transformando um *hobby* num negócio também deveria fazer parte desse ambiente. Aliás, pouco a pouco a percepção de que era possível fazer dinheiro vendendo microcomputadores e desenvolvendo *software* se tornará o modo de pensar dominante.

Foi ao final dos anos sessenta e início dos setenta, que brotaram alguns projetos que se tornaram famosos por seu pioneirismo. O *Resource One*, foi uma espécie de comunidade informática que se constituía numa base de dados urbanos acessível a todos, coletava todas

[6]No original: Over time, both these beliefs and the networks of readers and contributors who developed them, along with the Catalog itself, helped create the cultural conditions under which microcomputers and computer networks could be imagined as tools of liberation.

informações úteis às atividades comunitárias da região. Outro projeto desenvolvido foi o *Community Memory* utilizando uma rede de terminais espalhados por toda região e o seu objetivo era “uma democracia direta em temos de informação”. O sistema funcionava sem controle central sobre as informações que cada um podia produzir ou ler. De certa forma, algo similar ao que ocorre na internet atualmente, A *Community Memory* era apresentada como uma alternativa ao uso dominante das mídias eletrônicas que provocavam a passividade dos usuários (TORRES, 2013). Outra iniciativa foi o tablóide *People's Computer Company* (PCC) cujo conteúdo está diretamente ligada às ideias de que os computadores devem ser controlados pelas pessoas e não o contrário. Bob Albrecht - fundador do PCC - buscava a desmistificação do computador e a difusão da ideia de que computadores poderiam ser utilizados para aprendizagem. Algum tempo depois do lançamento do *People's Computer Company,* ele também fundou o *People's Compute O movimento Software Livre e a Iniciativa Open Sourcer Center* numa pequena sala comercial em que ofereceu terminais com ligação a um computador no qual as pessoas podiam entrar para programar ou jogar (TORRES, 2013; MARKOFF 2005).

Todos esses projetos relacionados anteriormente foram importantes e influentes para a comunidade *hacker* que havia se formado no *Silicon Valley.* As ideias coletivistas, a liberdade de conhecimento, compartilhar suas criações e melhorias fazia parte da ética *hacker*. Em 1975, um pequeno grupo de *hackers* fundou o *Homebrew Computer Club* em meio à euforia do lançamento do primeiro microcomputador vendido comercialmente nos Estados Unidos, o Altair 8800, que também foi comercializado em forma de kit para que pudesse ser montado em casa (COLEMAN, 2013).

Na primeira reunião do *Homebrew Computer Club* (HCC), realizada na garagem de um dos fundadores, compareceram cerca de trinta e dois entusiastas interessados em aprender mais sobre computadores e trocar experiências (TORRES, 2013). Seis dos que estavam presentes na primeira reunião já havia construído seus próprios computadores. A computação ainda estava presa em empresas e laboratórios de pesquisa, portanto a oportunidade de ver um computador funcionando não poderia ser desperdiçada. Nas reuniões do clube, as pessoas ficavam ao redor da sala fazendo suas apresentações e logo em seguida iniciavam a importante etapa de partilha de informação técnica. Esse instante de compartilhar a informação se tornou um marco da experiência *Homebrew.* O número de membros do grupo foi crescendo a cada reunião realizada, a tal ponto que, começou a publicar um boletim informativo, o qual continha, em sua maioria, textos técnicos. Porém, assuntos auxiliavam as pessoas que não faziam parte do clube a

compreender o funcionamento de computadores e também a produção de softwares dentro do ambiente do clube também foram encontrando espaço dentro dos boletins. Havia o entendimento de que a informação deveria ser disponível a todos, afinal uma das propostas do *Homebrew* era ser um espaço para compartilhar a informação e o conhecimento. Um exemplo dessa atuação política dos membros do clube, na carta de propósitos, o grupo estabeleceu que os *softwares* produzidos através do uso de equipamentos de propriedade do clube deveriam ser disponibilizados em domínio público.

O estado de espírito cooperativo dos membros do HCC era tamanho que os *hackers* que ajudavam a coordenar as reuniões incentivavam os desenvolvedores a trazerem para os encontros os seus programas de computador e os exortava a compartilhá-los com os outros componentes do clube. John Markoff comenta em seu livro que “desde o início, Felsenstein incentivou esta economia da dádiva[7], exortando os *hackers*, "Trazer de volta mais do que você tomar."[8]. (MARKOFF, 2005)

A maioria dos *hackers* do *Homebrew* não frequentava as reuniões em busca de aprimorar seus conhecimentos para aplicá-los no mercado e ganhar dinheiro vendendo licenças de *softwares*, pois na cultura do *hobby*, *software* não era negócio (MARKOFF, 2005). A ideia era aprimorar seus conhecimento e trocar experiências “por diversão”. A ética *hacker* era a base filosófica comum a todos dentro do *Homebrew*. Entretanto, o conceito original da palavra *hacker* nasceu no *Tech Model Railroad Club* (TMRC), um clube de admiradores de ferromodelismo fundado no *Massachusetts Institute of Technology* (MIT) durante os anos quarenta. Segundo Levy a palavra *hacker* possui um sentido de “[...] um projeto realizado ou um produto construído não apenas para cumprir um objetivo construtivo, mas com um pouco de prazer selvagem

[7]A economia da dádiva pode ser entendida como “tudo o que circula na sociedade que não está ligado nem ao mercado, nem ao Estado (redistribuição), nem à violência física. É o que circula em prol do ou em nome do laço social.” “Basta pensar no que circula entre amigos, entre vizinhos, entre parentes, sob a forma de presentes, de hospitalidade e de serviços. Na sociedade moderna, a dádiva circula também entre desconhecidos: doações de sangue, de órgãos, filantropia, doações humanitárias, benevolência.” Um dos comportamentos sui generis da dádiva é a negação da importância da dádiva por parte do doador. Mauss observa que o doador demonstra uma modéstia exagerada. No entanto, nosso próprio comportamento tem o mesmo sentido, visto que, desse modo, os receptores da dádiva diminuem a obrigação de retribuir e tornam a retribuição incerta. É uma maneira de tornar o outro livre para dar quando tiver vontade. Godbout explica da seguinte maneira: “se aquilo que se lhe deu não é nada, ele não fica obrigado a retribuir, fica livre para dar; e se der, será também uma dádiva de verdade. Dá-se assim ao receptor a possibilidade de fazer uma verdadeira dádiva, em vez de se conformar à obrigação de retribuir.” Chamo a atenção do leitor para uma sutil semelhança com a frase “Libere cedo, libere frequentemente” do famoso artigo “ A Catedral e o Bazar” de Eric Raymond, no qual ele relata a maneira de trabalho cooperado (e compartilhado) de Linus Torvalds durante o desenvolvimento do kernel do Linux. A frase se refere a disponibilizar para o público o produto do trabalho de desenvolvedores para as pessoas de uma comunidade utilizar, testar, melhorar, criticar.

[8]From the start, Felsenstein encouraged this gift economy, urging the hackers, “Bring back more than you take.”

envolvido, foi chamado de "*hack*."[9] Com o passar do tempo a palavra foi associado também a brincadeira ou trote. Ainda segunda Levy, o conceito de *hacker* está ligado à inovação, estilo e virtuosismo técnico. Por isso, Levy relacionava as características da Ética Hacker:

> 1. O acesso a computadores - e qualquer coisa que possa lhe ensinar algo sobre a maneira como o mundo funciona - deve ser ilimitado e total. Sempre produzir mãos na massa. Imperativo!
> 2. Toda a informação deve ser livre.
> 3. Desconfiança na autoridade - promover a descentralização.
> 4. Hackers devem ser julgados pela sua hacking, não critérios falsos como diplomas, idade, raça ou posição.
> 5. Você pode criar arte e beleza em um computador.
> 6. Os computadores podem mudar a sua vida para melhor.
>
> (LEVY, 2010 p.28-34 tradução nossa)[10]

O *Homebrew Computer Club* estava destinado a mudar o mundo, mas quando a mudança ocorreu, não foi aquela mudança libertária que alguns de seus fundadores esperavam (MARKOFF, 2005). O *Homebrew* acabou desempenhando uma papel de catalisador para a indústria de computadores pessoais e para a indústria do software. Um dos capitalistas de *venture capital*, John Doerr, referiu-se como "[…] a maior acumulação legal de dinheiro na história" (MARKOFF, 2005).

Um episódio fundamental, que ratifica a frase de John Doerr, envolveu a linguagem de programação de computadores BASIC concebida por Bill Gates e Paul Allen, dois jovens *hackers* que também participavam do *Homebrew,* fundadores da empresa Microsoft (que na época era conhecida como Micro-Soft). A linguagem foi incorporada aos computadores Altair, da empresa MITS, que era a sensação entre os *hackers* do HCC.

Segundo Levy, alguns *hackers* do *Homebrew* estavam descontes com o modelo de negócios adotado pela MITS, pois afirmavam empresa contrariava a ética *hacker*, visto que havia adotado uma atitude que prejudicava seus concorrentes. A conduta *hacker* entendia que as empresas deveriam compartilhar com todos, inclusive outras empresas, o seu plano de negócios e

[9]No original: […] a project undertaken or a product built not solely to fulfill some constructive goal, but with some wild pleasure taken in mere involvement, was called a "hack."

[10]Access to computers—and anything that might teach you something about the way the world works — should be unlimited and total. Always yield to the Hands-On Imperative!;
All information should be free;
Mistrust Authority—Promote Decentralization;
Hackers should be judged by their hacking, not bogus criteria such as degrees, age, race, or position;
You can create art and beauty on a computer;
Computers can change your life for the better.

informações técnicas para que coletivamente possam ser melhorados os *softwares* gerando ganhos para todos. Essa interpretação de uma parte dos *hackers* do *Homebrew* para com a maneira da empresa MITS conduzir seus negócios, descrita por Levy, demonstra a oposição dos *hackers* com aqueles que entendiam a computação como apenas um negócio. O surgimento de empresas nesse ambiente que unia atuação política com conhecimento técnico, já demostrava a percepção de oportunidade de negócios dos empreendedores e de capitalistas. Entrar nesse universo de apaixonados por tecnologia era a possibilidade de explorar um mercado de consumidores ainda pequeno, mas com muitas possibilidades de expansão. Estavam erguendo os alicerces do modelo de negócios adotado pela indústria de software ao final dos anos setenta.

Buscando atrair novos clientes para seu produto, o MITS, fez uma apresentação do Altair em que estava rodando uma versão do BASIC. Alguém (que até hoje não foi identificado) pegou "emprestado" uma cópia do BASIC e levou para *hacker* Dan Sokol fazer algumas cópias para membros do HCC (MARKOFF, 2005).

Segundo Levy, (p.232) "na próxima reunião *Homebrew Computer Club*, ele veio com uma caixa de fitas. Sokol cobrou o que em termos de *hackers* foi o preço adequado para o *software*: nada. A única condição era que se você pegasse uma fita, você deveria fazer cópias e vir para a próxima reunião, com duas fitas. E distribuí-las."[11] Os outros integrantes seguiram o exemplo e compraram as fitas e distribuíram para colegas do HCC além de outros grupos de *hackers* da região. Dessa maneira, a primeira versão do Altair BASIC foi compartilhada livremente antes mesmo de ser lançada oficialmente.

Entretanto, havia dois *hackers* que não estavam muito satisfeitos com essa movimentação em torno do BASIC. Paul Allen e Bill Gates desenvolveram o BASIC para o MITS com a intenção de lucrar com cada cópia vendida e não por que acreditavam que estavam contribuindo para um mundo melhor. Na verdade, eles se sentiram roubados. (LEVY, 2010). Bill Gates, então furioso com toda a situação, escreveu a famosa Carta Aberta aos Hobistas (*Open Letter to Hobbyists*) do *Homebrew Computer Club*, na qual condenava o ato daqueles que copiaram indevidamente o BASIC. "Como a maioria dos hobbistas devem saber, muitos de vocês roubaram o meu *software*. *Hardware* deve ser pago, mas *software* é algo para ser compartilhado. Quem se importa se as pessoas que trabalharam nele foram pagas?" Escreveu Gates. (LEVY, 2010)[12].

[11]No original: Sokol charged what in hacker terms was the proper price for software: nothing. The only stipulation was that if you took a tape, you should make copies and come to the next meeting with two tapes. And give them away.

[12]As the majority of hobbyists must be aware, most of you steal your software. Hardware must be paid for, but software is something to share. Who cares if the people who worked on it get paid?

Definitivamente, a ética *hacker* encontrava a lógica do mercado. Através dessa carta, segundo Coleman (2013), Gates e Allen, justificam sua posição com uma das razões utilitaristas mais comuns entre os defensores da propriedade intelectual. Para um "*software* de qualidade" ser escrito os autores devem receber algum incentivo financeiro e o *copyright* cumprem essa função, porque regula a reprodução de cópias desse *software*. De fato, essa carta contribuiu diretamente para a prática de transformar o software numa mercadoria de alto valor agregado e vendê-lo como se fosse um bem tangível.

Bill Gates recebeu muitas cartas de hobbistas ofendidos com a acusação de roubo e outras, em número bem menor, de apoio. Gates escreveu inclusive uma segunda carta na tentativa de elucidar alguns mal entendidos e ratificar o que já havia escrito na carta anterior. Não obstante Bill Gates não fez muito além de reclamar aos hobbistas por terem pirateado seu *software*. No entanto, essa mesma "pirataria" contribuiu para a disseminação do BASIC pelo mundo. Qualquer pessoa que possuía o Altair BASIC - ou que viesse adquirir um – tornou-se um potencial cliente da Microsoft. Aliás, ainda hoje, a Microsoft se queixa da pirataria de seus *softwares* alegando prejuízo financeiro, todavia também é a principal beneficiada dessa pirataria, dado que seus *softwares* se tornaram o padrão em praticamente todo mundo estabelecendo um monopólio de programas de computador..

Segundo Markoff (2005), este confronto entre Bill Gates e os *hackers* do *Homebrew* foi o antecedente da tensão que envolveu não apenas a indústria de computadores, mas também o mundo da música, indústria do entretenimento e o mundo editorial também. Um confronto do início da era do computador pessoal que hoje se tornou um dos mais profundos conflitos da economia mundial.

Se por um lado o Vale do Silício se mostrou o berço de movimentos que uniam contracultura, ativismo político, liberdade de expressão e paixão por computadores, por outro lado se salientou como um fonte de inúmeras empresas inovadoras. Muitas delas, na verdade, surgiram não muito tempo depois da primeira reunião do *Homebrew Computer Club* acontecer. Marcas valiosas e admiradas como Apple além da Osborne Computer, Cromemco e North Star.

Aracele Torres (2012), afirma que Microsoft soube reconhecer antes das demais empresas, "[...] que o *software* deveria ser o principal agente no desenvolvimento da computação e que isso não ocorreria se ele continuasse sendo compartilhado gratuitamente." Bill Gates havia reconhecido o que todos seus companheiros de clube não: que com o advento dos baratos computadores pessoais, o software poderia e deveria vir à tona como seu principal componente. E

somente através da cobrança de dinheiro por ele — mesmo que ele tenha sido originalmente livre — isso poderia acontecer. Em 1978 sua empresa, agora chamada “Microsoft”, tinha rompido relações com MITS e estava se mudando de Albuquerque para o subúrbio de Seattle (o próprio MITS tinha perdido sua identidade, estava sendo comprado por Pertec em 1977). Computadores estavam de fato ganhando espaço na vida das pessoas e se tornando umas mercadorias muito cobiçadas. Mas a força propulsora não foi a visão contracultural de uma utopia da informação livre e compartilhada; foi a oportunidade de criar um modelo de negócios que protegesse os proprietários dos *softwares* de cópias não autorizadas e os remunerasse a cada nova licença de uso vendida.

Muitos *hackers* assumiram um perfil profissional fundando empresas ou se tornando funcionário delas. Se antes o que os vinculava era a ética *hacker*, agora era a concorrência para ampliar e manter a fatia de mercado. A ética *hacker* começa a perder adeptos e divulgadores. Porém, poucos lembram que a origem de todos eles foi a cultura *hacker*. Quando Richard Stallman entra no MIT no início dos anos setenta, essa cultura estava, lentamente, sendo esquecida pelos *hackers*, os quais eram contratados para trabalharem como desenvolvedores em empresas de tecnologia e uma das condições para o trabalho era assinar contratos de confidencialidades em que os trabalhadores ficavam impedidos de partilhar informações e o produto de seu trabalho com outras pessoas. O *software* era licenciado de modo a não permitir que fossem compartilhados, mas comprados. Diante desse cenário é que nasce o Movimento Software Livre, como tentativa de fazer o retorno àquela ética perdida. Veremos a seguir como isso aconteceu e se é ou não possível o desenvolvimento de uma ideologia colaborativa não mercantil no coração do sistema capitalista, ou seja, até que ponto podem conviver software livre e indústria da informação ou se há uma tendência de uma destas subordinar a outra e, nesse caso, quem subordina quem.

3. O MOVIMENTO SOFTWARE LIVRE E A INICIATIVA OPEN SOURCE

Os momentos históricos narrados no capítulo anterior foram os antecedentes que moldaram do mercado de tecnologia tal qual conhecemos atualmente. Não obstante, durante os anos oitenta do século vinte, surge um movimento técnico-social que vai de encontro às práticas da indústria de TI, principalmente, pelo formato de licenciamento, o incentivo ao compartilhamento e retorno ao convívio comunitário, ainda que virtual.

3.1 UM MANIFESTO E UM MOVIMENTO SOCIAL

Para entendermos o movimento software livre precisamos, antes, compreendermos quem foi seu fundador e sua motivação. Quando Richard Stallman começou a trabalhar no laboratório de Inteligência Artificial do *Massachusetts Institute of Technology* - MIT – em 1971, como programador, ele passou a fazer parte da comunidade *hacker* em que compartilhar software e o conhecimento era um hábito que existia há muitos anos. Stallman, que também é conhecido por RMS, desenvolveu um editor chamado *Emacs*, o qual foi compartilhado livremente com todo aquele que concordava com uma condição: todas as melhorias feitas pelos usuários do *Emacs* deveriam ser compartilhadas também. Stallman batizou esse acordo informal de a comuna *Emacs*. (LEVY, 2010).

A comunidade *hacker* da qual Stallman fazia parte estava diminuindo, no início dos anos 80 época em que os computadores pessoais se tornaram populares, pois muitos de seus colegas estavam indo trabalhar em empresas cujo foco estava em prover soluções ao mercado consumidor de computadores pessoais.[13] Além disso, o MIT descontinuou um projeto que envolvia muitos programadores. No livro *Free Software, Free Society,* Stallman descreve o momento da seguinte maneira: “A comunidade *hacker* do AI *Lab* já havia entrado em colapso, não muito tempo antes. Em 1981, a empresa Symbolics havia contratado quase todos os *hackers* do AI *Lab* e a comunidade despovoada foi incapaz de manter-se. (STALLMAN, 2010 tradução nossa)[14]

Manter uma comunidade de ativistas *hackers* naquele momento era bastante difícil, porque muitos desenvolvedores assinavam contratos com cláusulas de confidencialidade nas

[13]Um esclarecimento oportuno ao leitor é que computador pessoal foi o nome dado ao computador desenvolvido pela IBM nos anos 80 para concorrer com empresas como a Apple. Esse nome ficou como sinônimo de computadores de mesa utilizados em empresas, universidades, residências.

[14]No original: The AI Lab hacker community had already collapsed, not long before. In 1981, the spin-off company Symbolics had hired away nearly all of the hackers from the AI Lab, and the depopulated community was unable to maintain itself.

empresas em que trabalhavam. Dessa maneira, eles ficavam impedidos de compartilhar informações com seus amigos e colegas sob pena de perderem o emprego e serem processados. O conhecimento era, de fato, um bem privado de propriedade de uma empresa. (STALLMAN, 2010)

Todo esse somatório de acontecimentos o forçou a tomar a decisão de se desligar do MIT e desenvolver um sistema operacional, o qual é um *software* fundamental para quem deseja utilizar um computador. Stallman entendia que ao iniciar esse projeto de desenvolvimento ele poderia voltar a ter uma comunidade de *hackers* colaborativa e atuante. (STALLMAN, 2010 p.9) Assim, ele anuncia em uma lista e e-mails que começará a desenvolver um software compatível com o sistema Unix[15] chamado GNU[16] e que irá distribuí-lo gratuitamente a todos que puderem usá-lo. A partir desse instante, o ambicioso projeto de desenvolver um *software* operacional com ajuda de trabalho voluntário e colaborativo começa a tomar forma.

Em 1985, Stallman funda a *Free Software Foundation* (FSF), a qual é uma organização sem fins lucrativos, para fazer a gestão do Projeto GNU e captar recursos (TORRES, 2013). Diferentemente da atuação das empresas de software proprietário, a FSF permitia que os usuários compartilhassem, modificassem e redistribuíssem o código-fonte do *software*. A FSF também vendia fitas (aquela era uma época anterior aos discos removíveis como o disquete) com o *software* a preços menores do que os praticados pelas empresas desenvolvedoras de *software* proprietário (COLEMAN, 2012). No entanto, o acordo entre as partes era informal, ou seja, não havia um modelo de licenciamento ou licenças. A evolução do Projeto GNU trouxe um problema adicional com o qual a FSF precisou enfrentar. Como garantir que o código-fonte do *software* original e suas melhorias permanecessem livres e sem correr o risco de que alguém pudesse registrá-lo sob uma licença *copyright* e, com isso, tornando-o um *software* proprietário para faturar vendendo licenças de uso? Em outras palavras, como garantir que o *software* permanecesse livre e acessível a todos no futuro? A licença de domínio público não garantiria, necessariamente, que o software e suas melhorias permaneceriam abertas. Stallman, precisou enfrentar uma disputa judicial sobre o direito autoral de seu editor de texto *Emacs*[17] para perceber

[15]Unix foi um sistema operacional para computadores, criado pelos programadores Ken Thompson e Dennis Ritchie que eram funcionários da AT&T. O software não era de código aberto, porque a AT&T detinha uma licença *copyright* dele, no entanto a empresa disponibilizava o código-fonte do Unix e permitia que ele fosse compartilhado apenas no ambiente acadêmico.

[16]O significado da sigla GNU é a expressão: *Gnu's Not Unix.* Quer dizer, traduzindo ao pé da letra, que o sistema GNU não é Unix. Trata-se de um acrônimo. É comum o uso de acrônimos pelos seguidores da cultura *hacker.*

[17]Torres detalha essa história em seu trabalho. “Richard Stallman ao desenvolver o Emacs para o GNU havia copiado parte da estrutura de outra versão do Emacs, construída por um programador chamado James Gosling. James havia

que era necessária a formalização de um mecanismo legal para a proteção dos programas de computador. Em 1989, Stallman, havia criado um enquadramento jurídico para o *software* livre, com vistas a evitar o problema que irrompeu sobre o seu trabalho no caso do editor de texto *Emacs* não se repita além de adicionar uma proteção para o software livre e, fundamentalmente, as liberdades dos usuários (COLEMAN, 2012). Stallman, "hackeia" o *copyright,* pois a GPL (*General Public Licence)* é uma licença que está construída em cima de direitos de autor, porém invertendo o seus princípios. Em vez de conferir ao proprietário de um direito de autor o poder de restringir cópias, o proprietário de um direito de autor concede aos usuários o direito de copiar e compartilhar programas de computador. E a GPL vai além, funcionando como uma proteção contra a ameaça futura de privatização das versões que ainda estão por vir de um *software*. Todo o programa de computador que for licenciado sob a GNU GPL deverá permanecer sob a mesma licença mesmo sofrendo modificações. Isso é diferente de registrar o *software* em domínio público, porque o conteúdo licenciado desta forma pode ser, posteriormente, retomado em um novo trabalho, que por sua vez pode ser protegido com alguma licença proprietária que fecha o código-fonte e proíbe o compartilhamento. (COLEMAN, 2012; TORRES 2013)

É de extraordinária perspicácia desenvolver uma licença que inverte o sistema de registro de direito do autor. O *copyright* é um sistema restritivo, pois não permite a cópia e nem a distribuição do bem sem que o autor consinta. A licença é de uso, pois qualquer tipo de alteração da obra poderá ser motivo de litígio entre o autor e quem copiou ou modificou sem autorização. A GPL garante a liberdade que o *copyright* restringe. Podemos pensar o *copyleft* como a antítese do *copyright* e essa é uma das razões pela qual levantou suspeitas de ser um conceito anticapitalista ou comunista[18]. Em outras palavras, o *copyleft* em termos legais é uma aplicação do *copyright*,

colocado a sua versão sob copyright e vendido os direitos para uma empresa chamada UniPress, que mais tarde ameaçou Stallman por infringir o copyright do GOSMACS (Gosling Emacs) ao lançar o seu GNU Emacs, Stallman teve que refazer o seu Emacs, suprimindo a parte copiada."

[18]No documentário Revolution OS, Michael Tiemann, um dos fundadores da Cygnus Solutions, a primeira empresa a oferecer software livre comercialmente, descreve um momento relativamente engraçado sobre a comparação do software livre com o comunismo em que um funcionário da Universidade de Moscou (essa história ocorreu em 1990, portanto ainda existia a URSS) visitou a Cygnus Solutions, por indicação de Richard Stallma, porque estava interessado em entender como funcionava o modelo do software livre para implantar na Rússia e fomentar inovações naquele país. Tiemann conta que conforme foi apresentando o plano comercial ao funcionário da universidade mais ele balançava a cabeça de um lado para outro fazendo um gesto negativo. Então, Tiemann perguntou o que havia de errado no material que ele estava apresentando. Então, o russo o respondeu: "Isso soa muito com o comunismo para ter sucesso na Rússia." Um momento jocoso mas que nos oportuniza uma rápida especulação do porquê da frase: a) Pode que ser que o funcionário não possuía muitas condições de avaliar o que estava diante dele e confundiu alhos com bugalhos; b) O funcionário russo estava a serviço de um governo de transição do socialismo real para o capitalismo, que foi ser implantado em 1991, e, por isso, achou a ideia demasiadamente subversiva para o momento em que o país vivia; c) Que o conjunto de ideias e métodos do software livre é de ruptura não apenas com o capitalismo, mas também com o modelo de socialismo real que existia na Rússia e no Leste Europeu.

não uma negação. Em termos morais significa decidir não ser o guardião do uso que outras pessoas fazem do seu trabalho, não usar o poder derivado da propriedade, exceto para impedir que alguma outra pessoa, sem considerar o próprio autor ou o público, imponha este poder. Stallman descreve o momento em que a palavra *copyleft* foi criada em uma nota de rodapé em seu livro "*Free Software, Free Society"*, no entanto essa pequena nota nos permite entender o significado filosófico e político do nome, vejamos:

> Em 1984 ou 1985, Don Hopkins (um sujeito muito imaginativo) enviou-me uma carta. No envelope que tinha escrito vários provérbios divertidos, incluindo este: 'Copyleft - todos os direitos revertidos.' Eu usei a palavra 'copyleft' para nomear o conceito de distribuição eu estava desenvolvendo no momento. (STALLMAN, 2010 p. 12. tradução nossa)[19]

3.2 A EXPANSÃO DO SOFTWARE LIVRE E O SURGIMENTO DO LINUX

Apesar de alguma controvérsia sobre a nomenclatura utilizada pela FSF, Richard Stallman continuava como líder do Projeto GNU desenvolvendo outros softwares para compôr o sistema operacional GNU, no entanto faltava a criação de seu núcleo (kernel)[20]. O núcleo do GNU, o Hurd, apresentava uma arquitetura elegante, porém complexa, o que tornava seu desenvolvimento difícil e mais demorado do que Stallman projetara.

Em 1991, o programador Linus Torvalds, estudante no Departamento de Ciências da Computação da Universidade de Helsinki, Finlândia, havia começado a escrever um núcleo baseado no Minix, um sistema educacional escrito pelo programador Andrew Tanenbaum e baseado no Unix (TORRES, 2013; CARMONA, 2008). O sistema que Torvalds projetou foi um projeto pessoal, algo como um *hobby,* e o seu código-fonte foi disponibilizado na internet junto com um pedido de ajuda a outros programadores para que sugerissem o que deveria ter ou não nele. Linux, como foi chamado o *kernel* desenvolvido por Linus Torvalds, usava a licença GPL e em 1992 foi incorporado ao sistema GNU, gerando um sistema operacional livre e completo denominado GNU/Linux, embora hoje seja popularmente conhecido apenas como Linux.

---

[19]No original: In 1984 or 1985, Don Hopkins (a very imaginative fellow) mailed me a letter. On the envelope he had written several amusing sayings, including this one: "Copyleft — all rights reversed." I used the word "copyleft" to name the distribution concept I was developing at the time.

[20] O kernel é uma parte central do sistema, responsável pela configuração e gerenciamento dos dispositivos (teclado, mouse, monitor, etc).

Com o lançamento do sistema operacional GNU/Linux as atenções do mundo da tecnologia se voltaram para ele. As pessoas especulavam sobre sua qualidade técnica e tentavam entender como era possível um sistema de computador completo ser produzido fora de um ambiente empresarial, através de trabalho coletivo e ser um software livre que poderia ser baixado diretamente da internet e instalado a um custo quase zero[21], uma vez que não há necessidade de pagar por uma licença. Sua arquitetura aberta e sua comunidade de contribuidores o transformara, em pouco tempo, em um sistema operacional estável equivalente aos sistemas operacionais privados e restritos que há muito já estavam consolidados e competindo pelo mercado consumidor.

3.3 ELOGIO AO MÉTODO

A aparição (e consolidação) meteórica do Linux está ligada ao método de desenvolvimento inovador e participativo usado por Torvalds. Em 1997, Eric Raymond o apresenta através do artigo “A Catedral e o Bazar”, o qual descreve o método aberto e decentralizado de desenvolvimento o comparando a um “bazar”. O modelo “bazar” seria o contraponto ao modelo tradicional de desenvolvimento de *software* chamado por Raymond de “catedral”. A crítica feita no artigo é voltada ao modelo de desenvolvimento proprietário, mas também fazia referência ao método de desenvolvimento adotado pela *Free Software Fundation*. Ao indicar que até surgir o Linux, os códigos eram como se fossem "catedrais", “monumentos sólidos construídos a partir de um planejamento central” (EVANGELISTA 2010).

Eric Raymond aponta muitas virtudes ao método desenvolvido por Torvalds, principalmente, a frequente liberação das alterações feitas no código-fonte. Dessa forma, programadores de qualquer parte do mundo – com acesso à internet - teriam condições de avaliar as alterações, testá-las e melhorá-las – se fosse o caso – devolvendo as modificações ao Torvalds. Diz Raymond:

> Aqui, eu penso, é o centro da diferença fundamental entre os estilos bazar e catedral. Na visão catedral de programação, erros e problemas de desenvolvimento são difíceis, insidiosos, um fenômeno profundo. Levam meses de exame minucioso por poucas pessoas dedicadas para desenvolver a confiança de que você se livrou de todos eles. Por

[21] O custo quase zero se refere ao custo de contratar técnicos para realizar a instalação já que nem todas as pessoas dominam o processo de instalação de um sistema operacional.

> conseguinte os longos intervalos de liberação, e o inevitável desapontamento quando as liberações, por tanto tempo esperadas, não são perfeitas. Na visão bazar, por outro lado, você assume que erros são geralmente um fenômeno trivial – ou, pelo menos, eles se tornam triviais muito rapidamente quando expostos para centenas de ávidos co-desenvolvedores triturando cada nova liberação. Consequentemente, você libera frequentemente para ter mais correções, e como um benéfico efeito colateral você tem menos a perder se um erro ocasional aparece. (Raymond. p. 7-8, 1997)

Esse é um documento importante em que as ideias de cooperação, compartilhamento de software e de conhecimento são apontadas como positivos para a sociedade. No entanto, há mais do que o elogio ao método de Torvalds no artigo escrito por Raymond, como aponta Evangelista:

> Mas há mais no que diz Raymond com relação ao modelo Linux do que o elogio da astúcia e da técnica – embora o sucesso desta seja inegável – há uma disputa de poder sobre quem representa e o que significa o movimento. Stallman sempre foi uma figura politicamente muito atuante, não apenas no campo da informática. Mais velho, tendo vivido toda a experiência da luta pelos direitos civis nos EUA, Stallman carrega em sua fala críticas não muito ao gosto das empresas, em especial um conjunto de empresas da Califórnia que está tentando transformar o Linux em negócio. No site pessoal de Stallman, por exemplo, ao lado de artigos em favor do software livre encontram-se também ensaios políticos sobre temas como a invasão estadunidense ao Iraque e o muro de Israel na Palestina. Raymond, por sua vez, é um ardoroso defensor da liberalização do uso de armas, tema usualmente mais ligado às bandeiras da direita estadunidense (os conservadores). (EVANGELISTA, 2010 p.43)

Esse tipo de postura anuncia mais do que uma luta por poder entre os membros da comunidade software livre. Mostra, também, o quanto é politicamente heterogênea a maneira de seus membros pensarem. Para programadores que começaram a trabalhar durante os anos de consolidação do neoliberalismo, as ideias defendidas por Stallman soavam demasiadamente politizadas e radiciais. Programadores como o próprio Torvalds não partilham dos mesmas ideias políticos de Stallman. Podemos especular se é por que a ideia de “não existe almoço grátis” foi introjetada na cultura e faz parte de um modo de pensar e agir, por que não há um interesse (ou consciência) em lutar por uma sociedade mais justa ou por que, no fundo, muitos programadores sonham secretamente em montar uma empresa tão lucrativa como a Microsoft e Apple. Se o discurso do Stallman não agrada a certos grupos de programadores, ele é muito menos agradável aos empresários e empreendedores. Na internet, muito se fala e comenta sobre os benefícios que podem trazer para empresas, governo e sociedade a abertura e o compartilhamento. Raymond foi um grande contribuinte para a difusão dessa ideia ao descrever o inovador, descentralizado e evolutivo processo de produção do Linux, no qual, uma comunidade de usuários se torna co-produtores e auxiliam voluntariamente o desenvolvimento de um programa entregando rapidamente o software acabado e melhorado para a comunidade estabelecendo uma espécie de

“[...] em que as melhorias sobrevivem e as soluções falhas são logo identificadas”. (EVANGELISTA, 2010 p.44). Essa ideia foi decisiva no episódio envolvendo a Netscape, proprietária de um navegador de Internet que havia se tornado comercialmente popular, mas que foi alvo de uma ação agressiva anticompetitiva - segundo tribunais dos EUA – da Microsoft cujo objetivo era acabar com o mercado consumidor dos navegadores Netscape e garantir consumidores para seu Internet Explorer. Diz Evangelista:

> “Em 1998, Raymond foi a peça chave no processo de convencimento dos executivos da Netscape para que usassem uma licença livre para o navegador – então comercialmente morto – de modo que a comunidade continuasse seu desenvolvimento.” (EVANGELISTA, 2010 p.45).

A liberação do código-fonte do navegador de internet Netscape deu origem ao Mozilla Firefox. Esse evento aumentou o reconhecimento que Raymond já havia conquistado ao escrever o artigo A Catedral e o Bazar, porque a Netscape foi a primeira grande empresa a aderir a uma licença livre (TORRES, 2013). O movimento Software Livre obteve um grande acréscimo de popularidade através do surgimento do Mozilla Firefox e do Linux cuja fama o fez perder a palavra GNU. Na verdade, o nome correto é GNU/Linux, entretanto o nome Linux é que ficou popular para o desgosto do Stallman e dos seguidores da FSF que, sempre, precisam explicar o que é o GNU/Linux. Dizer Linux ou GNU/Linux também pode demarcar uma orientação política, pois desde a criação da Open Source Initiative, o movimento sofreu um racha que deu origem a dois grupos: o grupo *open* e o grupo *free*.

3.4 O PÓS-LINUX, A OPEN SOURCE INITIATIVE E A DIVISÃO POLÍTICO-IDEOLÓGICA

Vários episódios contribuíram para a maturidade do Movimento Software Livre. Em 1998, Eric Raymond e Linus Torvalds foram protagonistas da fundação da *Open Source Initiative* (OSI), a qual focalizava que a adoção de software livre deveria ser feita por critérios técnicos e indicava o uso da expressão *“open source”* ao invés de *“free software”* rompendo o vínculo com a fardo ideológico que a expressão utilizada pela FSF apresentava. (KON et al 2009). No artigo *Goodbye, "free software"; hello, "open source"*, publicado em 1998, Raymond aponta que o principal motivo para usar o termo “*open source*” foi o de apresentar o *software* livre de uma

maneira mais agradável aos capitalistas que representam o mundo dos negócios. Ademais, utilizar a palavra "*open*" minimiza a imprecisão da palavra "*free*" cujos significados podem ser distintos em língua inglesa. De forma similar ao que acontece na língua portuguesa com a palavra livre, na língua inglesa a palavra "*free*" pode conter o sentido de liberdade ou de gratuidade.

Não há diferenças fundamentais entre as definições de *free software* e *open source*. Na prática, a OSI defende que o software deve respeitar as quatro liberdades elementares que a FSF enunciou e os detalhes técnicos que uma licença de *software* deve possuir para ser considerada livre ou não. Contudo, a diferença entre os discursos e a maior permissividade de licenciamento pela OSI colocam o *software* livre e o *software* de código aberto em posições distintas, por vezes. Pode-se afirmar que a FSF reconhece várias licenças de software livre, mas nem sempre as recomendam por não as considerarem compatíveis com o conceito de *copyleft*, o qual determina que qualquer um que distribui o software, com ou sem modificações, tem que passar adiante a liberdade de copiar e modificar novamente o programa. Dito de outra forma, a licença GPL (chamada de licença de reciprocidade total) determinam que qualquer trabalho derivado precisa ser distribuído sob os mesmos termos da licença original. A ideia do *copyleft* é permitir que todos possam executar, copiar, modificar e distribuir versões modificadas do programa, mas impedir que sejam adicionadas restrições a essas versões redistribuídas. Tal ideia visa fortalecer o software livre como um todo, não permitindo que melhorias do software sejam retiradas do alcance da comunidade. O resultado esperado é que a quantidade de *software* livre aumente cada vez mais, beneficiando todos os envolvidos na cadeia produtiva do *software* livre. (KON e SABINO 2009). Porém esse comportamento "viral" da GPL é alvo de críticas da indústria do *software*, porque essas empresas não podem copiar nem uma linha de código de um programa de computador licenciado sob a GPL sem que o produto derivado também se torne um software livre sob a mesma licença. Essa situação não e bem vista pelos capitalistas da indústria tecnológica, principalmente, porque a propriedade do *software* estaria ao alcance de todos. Esse dispositivo também começou a sofrer críticas da membros da OSI, como Eric Raymond, conforme aponta Evangelista:

> Muitos dos membros do grupo *open* (Raymond, inclusive) defendem atualmente modelos mais livres de licenciamento do que a GPL, semelhantes ao domínio público, afirmando que restrições como o efeito *copyleft* impedem uma maior adoção pelas empresas, que poderiam fazer o software evoluir ainda mais. Tanto a propriedade intelectual do software proprietário como direito autoral em sua forma "livre, mas com restrições colaborativas" obstaculizam. O primeiro porque exige tarifas para que a tecnologia circule, outro porque requer uma espécie de pedágio de reciprocidade, o compartilhamento da melhoria implementada de maneira que se torne não exclusivo. (EVANGELISTA, 2010 p.59)

Expressando as divergências políticas, Stallman escreveu um artigo chamado *"Why Open Source misses the point of Free Software"* esclarecendo alguns pontos divergentes mas também convergentes entre as ideias defendidas pela FSF e pela OSI. Stallman, sublinha que "[…] todo *software* livre existente se qualificaria como *open source*. Quase todos os *softwares* de código aberto são um *software* livre, mas há exceções. Primeiro, algumas licenças de código aberto são muito restritivas, assim elas não se qualificam como licenças livres."(STALLMAN, 2015) [22]Para Stallman, o *open source* se aproxima mais de ser uma metodologia de desenvolvimento preocupada em tornar o *software* "melhor" e o *software* livre é um movimento social que considera as liberdades do usuários um imperativo ético e a existência do *software* privativo é um problema social. Porém, ele reforça a ideia de que as duas entidades não são inimigas e o que o verdadeiro inimigo é o *software* privativo (não livre).

A ascensão do grupo *open* adicionou um pouco de pragmatismo num movimento social já pragmático na medida em que buscou se harmonizar com o capital. O pragmatismo é percebido quando ocorre um distanciamento dos princípios filosóficos mais puritanos para, em seu lugar, adquirir um perfil mais mercantil. Os mesmos empreendedores capitalistas que percebiam o *software* livre como hostil, começaram a perceber o *open source* como simpático. Essa visão facilitou a penetração de empreendedores dentro desse universo e de desenvolvedores dos projetos *open source* implementando novas práticas dentro das empresas capitalistas. Com o avanço das empresas que trabalham dentro da lógica do *software* livre/aberto mais desenvolvedores puderam trabalhar de forma remunerada com o que antes era apenas trabalho voluntário. Sem falar na aparição de todo um ecossistema com eventos voltados para esse público, entidades sem fins lucrativos que promovem a sua difusão, o seu desenvolvimento e a sua consolidação.

No que diz respeito ao ganho para as empresas tradicionais de TIC, pode-se afirmar que o *software* livre/aberto é um novo nicho para exploração comercial e com um grande potencial de faturamento. A segunda é que essas mesmas empresas, com seus segredos industriais protegidos por direitos autorais, agora, pouco a pouco se tornam mais abertas às ideias de colaboração através da internet e compartilhamento[23], inclusive de conteúdo, formando grupos de

[22]No original: "[…] all existing free software would qualify as open source. Nearly all open source software is free software, but there are exceptions. First, some open source licenses are too restrictive, so they do not qualify as free licenses."

[23]Um exemplo interessante está no artigo "Entendendo o 'novo poder'" escrito por Jeremy Heimans e Henry Timms. Dizem os autores: uma revista pede aos seus leitores que renovem suas assinaturas, um fabricante pede aos clientes que comprem seus sapatos. Mas esse o novo poder aproveita a capacidade – e o desejo – crescente das pessoas de

consumidores que se tornam, eventualmente, co-desenvolvedores.[24] Não obstante, aos olhos de um observador atento, pode-se perceber que o *open source* está próximo de ser um modelo híbrido entre o que defende a FSF e o que o tradicional mercado de produtores e consumidores de TIC pratica e doutrina. Ainda que, do ponto de vista do licenciamento da propriedade intelectual, ambos os movimentos são similares.

3.5 A PROPRIEDADE INTELECTUAL NO CAPITALISMO CONTEMPORÂNEO

Usualmente, o que se classifica como propriedade intelectual é na verdade um conjunto de sistemas de proteção. Os elementos mais importantes desse conjunto são as marcas registradas, as patentes e o direito autoral (SIMON; VIEIRA, 2007). As formas de proteção usualmente utilizadas para o *software* são direito autorais e patentes. Colocando de lado a precisão conceitual jurídica, podemos afirmar que patente é uma concessão pública, conferida pelo Estado, que garante ao seu titular a exclusividade ao explorar comercialmente a sua criação. Em contrapartida, é disponibilizado acesso ao público sobre o conhecimento dos pontos essenciais e as reivindicações que caracterizam a novidade no invento. O direito autoral é o direito exclusivo de utilização, publicação ou reprodução de uma obra. Pode-se dizer que as patentes estão para as invenções assim como os direitos autorais estão para as criações artísticas (CARMONA, 2008).

A discussão que propomos nesse trabalho está centrada na questão dos direitos de autor. Toda discussão em torno do *software* livre, mais cedo ou mais tarde, leva-nos a questão do direito autoral e também das patentes de *software*, porque são mecanismos jurídicos utilizados para proteger o "segredo do negócio" e manter uma escassez artificial transformando um bem imaterial em material. A propriedade intelectual é um caso específico da propriedade privada com escopo de organizar e garantir o direito de uso e disposição de um bem que é desprovido de materialidade e é criado através de um trabalho imaterial. A legitimidade social da propriedade privada é essencial para o funcionamento do capitalismo, pois é a partir dela que se montam as

participar de forma que vão além do consumo. Esses comportamentos, […] incluem o compartilhamento (pegar o conteúdo de outra pessoa e compartilhá-lo com o público), a modelagem (remixar ou adaptar conteúdos ou recursos existentes com uma nova mensagem ou gosto), o financiamento (endossar com dinheiro), a produção (criar conteúdo ou fornecer produtos e serviços dentro de uma comunidade de pares, como o YouTube, Etsy ou Airbnb) e a copropriedade (como se vê na Wikipedia ou em softwares de código aberto).

[24]Um autor que procura capturar esse espírito do tempo é Don Tapscott através do best seller Wikinomics. No livro, Tapscott, enumera alguns exemplos de empresas que estão se "abrindo" para uma comunidade de consumidores que contribuem para a evolução de produtos e serviços.

relações sociais de produção. O fato de que de um lado estejam proprietários do capital (meios de produção em geral) e do outro os trabalhadores, proprietários exclusivos de força de trabalho, só ganha legitimidade se se passa a ideia de quem com muito trabalho se pode produzir capital. A propriedade privada se legitima sobre o interesse de todo individuo de fazer o seu esforço separável do esforço coletivo. Partindo-se ainda do princípio que certos recursos são escassos e que seu uso por alguém exclui o uso de outro, conduz-se ao raciocínio de que os direitos de uso devem ser cedidos aos que primeiramente os desenvolveram (ou descobriram, no caso de recursos naturais).

No entanto, se acompanharmos a organização social humana, em diferentes períodos históricos, veremos como a propriedade comum já foi útil para a expansão da riqueza e a legitimação social e política de regimes distintos do capitalismo. Daí Simon e Vieira (2007) falarem dos *commons* do passado. Segundo os autores:

> *Commons* são conjuntos de recursos utilizados em comum por uma determinada comunidade. Todos os membros dessa comunidade podem utilizá-los, de forma transparente, sem necessidade de permissões de acesso. Por outro lado, em commons bem-sucedidos, convencionam-se regras de uso responsável para que os recursos não se extingam ou se deteriorem. Historicamente (e em especial na Europa), os commons típicos eram as áreas de pasto, florestas, faixas litorâneas e outros recursos naturais utilizados cooperativamente, sem que um indivíduo fosse o único proprietário dos mesmos. São commons também os recursos ambientais compartilhados, tais como ar e água, e alguns bens essenciais para a vida nas cidades: ruas, parques, pontes etc. (Simon e Vieira 2007)

Áreas comunais de recursos materiais são, naturalmente, esgotáveis. Um exemplo clássico é o de uma rodovia. Se utilizarmos frequentemente, ela se deteriora num tempo menor; outro exemplo é se um pescador retira certa quantidade de peixes de um rio, esses peixes não estarão disponíveis para outros pescadores. “Ou seja, o uso simultâneo dos recursos materiais obedece a limites claros. Assim, os *commons* materiais são chamados de extinguíveis, competitivos ou rivais: no sentido de que o meu uso de um recurso rivaliza com o seu uso.” (Simon e Vieira 2007)

Os recursos materiais comunitários foram paulatinamente sendo desacreditados pelas próprias comunidades como resultado de um ataque sistemático ao comum e ao público. Na origem da argumentação a favor das áreas privadas está o artigo *“Tragedy of The Commons”*, do biólogo Garrett Hardin, publicado em 1968. Para ele, os *commons* sempre tenderiam à deterioração completa, pois a sua natureza permissiva conduziria a uso irresponsável e excessivo. Não são poucos os pesquisadores que se dedicaram a apontar as falhas do trabalho de Hardin indicando que ele excluiu de sua análise alguns tipos de *commons* socialmente geridos e que os

agentes só agem de forma egoísta visando satisfazer suas necessidades imediatas. Não podemos deixar de lembrar que essa análise é centrada na materialidade dos objetos e na escassez dos espaços tais como pastos, rios, lagos.

O conjunto de leis que regem a propriedade material e os bens tangíveis é diferente daquela que rege a propriedade imaterial e os bens intangíveis. No entanto, a propriedade imaterial recebe um tratamento similar ao da propriedade material. Porém, o debate sobre os direitos da propriedade intelectual está disposto, principalmente, sob duas linhas de pensamentos sobre o direito de propriedade que são: a propriedade intelectual natural ou a propriedade intelectual social. Há os que compreendem que existe um direito natural sobre os bens do trabalho imaterial, portanto, há de existir mecanismo que protejam o autor e sua obra. Assim, a propriedade intelectual não teria caráter técnico ou social, mas seria o reconhecimento de um direito absoluto do autor. Essa proteção funcionaria, também, como um incentivo a produção de bens intelectuais.

Por outro lado, há quem defenda que sistemas de proteção da propriedade são artificialmente construídos pela sociedade e devem ser limitados de forma a cumprir apenas a finalidade para qual foram concebidos (Simon e Vieira 2007). Por esse enfoque, a propriedade intelectual cumpre uma função social de promoção das artes e das ciências.

Outro ponto desse debate são os bens não-rivais e custo nulo de reprodução. Toda a produção científica e cultural – *commons* intelectual - foi possível graças a sua natureza não-rival. Os bens intelectuais são, ao mesmo tempo, produto final e matéria-prima. As invenções da imprensa, dos tocadores de discos, do cinema, da televisão contribuíram para a difusão de obras culturais e científicas. Caso a informação (científica ou cultural) se esgotasse com cada um desses usos, toda produção de bens intelectuais teria sido minimizada ao extremo (Simon e Vieira 2007). Atualmente, o desenvolvimento tecnológico possibilitou uma distribuição dos bens intelectuais, através da internet, num formato digitalizado com os custos de armazenamento e reprodução desprezíveis. Para um autor, o maior investimento é trabalho criativo de produção e desenvolvimento de sua obra, porque para distribuí-la, em certas situações, bastam alguns minutos para disponibilizá-las na internet. Ou seja, há alguns anos poderíamos afirmar que um filme numa fita VHS ou num DVD era um bem rival por mais que seu conteúdo não o fosse.

O avanço tecnológico fez com que a escassez artificial dos bens intangíveis se desintegrasse a ponto da indústria do entretenimento e, em seguida, a indústria de *software* desenvolvessem dispositivos tecnológicos[25] que bloqueassem o compartilhamento de músicas,

[25]A gestão de direitos digitais (em inglês *Digital Rights Management* ou *DRM*) consiste em delimitar a disseminação,

filmes, *softwares*. É perceptível a desvantagem trazida pela propriedade intelectual que torna os bens intangíveis, naturalmente abundantes, em escassos e rivais artificialmente. A tentativa de frear o compartilhamento também tem a intenção de tornar desinteressante o trabalho colaborativo feito através da internet. Conforme afirma Carmona (2008) "[...] sob o ponto de vista da propriedade intelectual, certamente o *copyleft* é uma das maiores, senão a maior, inovações nessa área desde a concepção do *copyright*." O *copyleft* serviu para clarificar o papel do *commons* na produção de bens intangíveis. De certa forma, foi o mecanismo que colocou "em evidência a importância do *trade-off* entre a propriedade intelectual e o *commons"* (Simon e Vieira 2007).

Em 2001, foi lançada uma nova licença que transbordou o escopo de copyleft que a GPL contém. O projeto Creative Commons é uma ferramenta com uma aplicação mais ampla para quaisquer produtores de conteúdo de outras áreas do conhecimento como as artes e ciências. As licenças do tipo Creative Commons (ou apenas CC) são configuráveis isso equivale a dizer que "a única característica fixa em todas as licenças é a exigência de atribuição (nos casos de citação, redistribuição ou derivação)" (CARMONA 2008). Licenças mais flexíveis são mais permeáveis para um maior envolvimento comunitário, porém diminuem o vigor da contribuição ao *commons* intelectual. Licenças mais "livres" (ou seja, mais alinhadas ao conceito do *copyleft*) contribuem mais ao *commons* a cada novo licenciamento; mas também podem atrair menos interessados, já que reservam menos direitos ao produtor, por um lado, e estabelecem algumas exclusões, por outro.

> Há aí um paradoxo. As licenças mais "livres", baseadas em copyleft (como a GPL ealgumas das licenças Creative Commons), são em certo sentido também mais restritivas: permitem qualquer uso livremente, mas exigem que as derivações sigam sempre determinadas condições. Assim, como já dissemos, embora a derivações de tais produtos seja permitida, ela não é irrestritamente livre, como no caso do domínio público (em que existe liberdade até mesmo para apropriar-se das derivações, restringindo seu acesso pelo direito autoral). Podemos ver essas licenças como os "contratos sociais" das comunidades formadas em torno da produção e compartilhamento de seus produtos. (SIMON; VIEIRA 2007)

A novidade subversiva desenvolvida por Stallman ainda nos anos 90 ao lançar a GPL e desenvolver o conceito de *copyleft* ainda permanece uma novidade. Há movimentos interessantes que apontam para o fortalecimento dessas iniciativas, porém ainda existem movimentos de resistências do "*establishment*". Como transpor esses movimentos e assegurar a sustentabilidade

---

por cópia, de conteúdos digitais, ao mesmo tempo em que se asseguram os direitos autorais e suas marcas registradas, pelo do proprietário dos direitos autorais. O objetivo do DRM é poder controlar um determinado conteúdo de maneira mais restrita.

do *software* livre apesar do modo de produção capitalista é o que investigaremos no próximo capítulo.

4. O COPYLEFT E A INDÚSTRIA DA INFORMAÇÃO: UMA CONVIVÊNCIA POSSÍVEL?

Anteriormente, apresentamos a indústria do software e o surgimento do *software* livre de maneira que fosse mais simples para o entendimento da pergunta que pretendemos responder nesse trabalho, qual seja: como é possível que a indústria da informação (a qual promove a informação como valor de troca) coexista com as ideias do *software* livre a seu formato de licenciamento? A relação entre o *copyright* (representante da indústria) e o *copyleft* (representando do software livre) será de subordinação ou de emancipação? É possível a existência de um modelo híbrido no qual a ideologia de colaboração e liberdade do *software* livre sobreviva, e até mesmo cresça, se aderir a certas práticas de mercado? Para começar a responder a esses questionamentos deveremos começar pela exploração dos modelos de negócios do *software* proprietário e *software* livre.

4.1 A LÓGICA DA PRODUÇÃO E DO COMÉRCIO DE *SOFTWARES*

Como vimos no primeiro capítulo, o *software* conquistou uma condição de indústria ao final dos anos 70 e início dos 80 do século XX. A maneira com a qual as empresas produtoras de programas de computador encontraram para lucrar foi restringindo o acesso ao código-fonte e proibindo cópias através de patentes e direitos de autor. Os formatos de comercialização podem ser condensados das seguintes maneiras: vender uma licença que permite apenas uma única pessoa utilizar o *software*, ou uma licença que permite várias pessoas utilizarem dentro de uma mesma organização; vender uma licença para a utilização de *software* de sustentação a um projeto em particular com diferentes preços e condições do que vender a licença para utilizar o mesmo *software* para uma finalidade diferente.[26] Em essência, esses são diferenciais de preços e adaptação de produtos para diferentes tipos de clientes. Além da licença de uso, o consumidor também pode estar interessado em adquirir suporte, manutenção, consultoria, serviços de

[26]Um exemplo dessa situação é a seguinte: uma empresa X resolve desenvolver um projeto especial voltado para educação. A empresa planeja vender um de seus *softwares*, uma suíte de escritório, por exemplo, com preço diferenciado para pessoas da área da educação. Porém se uma pessoa de outra área quisesse comprar uma licença do *software* pagaria o preço de balcão. Somente pessoas ligadas à educação pagariam o preço com desconto. Com essa estratégia a empresa poderia fidelizar educadores e tornar seu *software* muito mais conhecido e consolidado dentro do mercado.

integração entre sistemas e treinamento para capacitar os usuários a utilizar adequadamente o software (WEBER 2010, VÄLIMÄKI, 2005).

Para os consumidores, a necessidade de integração entre sistemas e o aprisionamento tecnológicos promovidos pela indústria de software proprietário causam problemas de toda ordem dentro das organizações. A lógica da indústria da tecnologia da informação criou um mercado, no qual o poder está mais nas mãos dos fornecedores do que com os clientes. Os desenvolvedores de *software* procuram atender aos seus clientes. No entanto "[...] a função de demanda é complexa, fragmentada, de rápida mutação e altamente granular, pois depende das circunstâncias muito específicas de usuários específicos" (WEBER, 2010 p.192). Assim, as empresas desenvolvedoras produzem bens levando em conta a demanda de clientes e produtos concorrentes. Contudo o resultado é, muitas vezes, uma solução sem as qualidades que os usuários julgam importantes. Os usuários, portanto, estão presos ao problema de tentar usar e integrar softwares relativamente inferiores a sua necessidade. Isso sem comentar a possibilidade de que sejam necessários novas correções e atualizações que a empresa desenvolvedora poderá demorar muito tempo para implementar. Isso porque apenas uma empresa é proprietária do código-fonte desse *software* e, portanto, apenas ela poderá corrigir atualizar e aprimorar esse *software.* Assim, as empresas que trabalham dentro dessa lógica conseguem construir um aprisionamento tecnológico do usuário através da escassez artificial que o *copyright* e as patentes de *software* permitem.

Entretanto, o mesmo dificilmente ocorre com o *software* livre, justamente, por sua natureza livre, aberta, colaborativa e acessível. Deter o controle da propriedade do código-fonte (e também da documentação) e ter uma grande comunidade que contribui com trabalho voluntário em nome de uma causa é os segredos desse negócio. A natureza não-rival do *software* necessita de uma proteção restritiva para que exista a necessidade de que cada usuário compre uma licença de uso e sem acesso ao código-fonte. A internet conseguiu a mágica de ser um meio de produção infinita de cópias de um programa de computador (sem falar em músicas, livros, filmes). Porém, pessoas trabalharam para que existisse a possibilidade de compartilhar, sem limites, esses programas. Como garantir que os desenvolvedores terão motivação suficiente para continuarem seu trabalho? Steaven Weber (2010) argumenta que ele não é apenas não-rival. Ele é um bem anti-rival. O meu uso do software livre não apenas não rivaliza com o uso de outra pessoa; pelo contrário, cada pessoa a mais que o usa auxilia os demais usuários. Diz Weber:

> *Software* em muitas circunstâncias é mais do que simplesmente não-rival. Sistemas operacionais como o Linux em particular e a maioria dos softwares em geral, na verdade, estão sujeitos a externalidades positivas de rede. Chamamos de um bem de rede, ou um bem anti-rival (um termo estranho, mas bem descritivo). Em linguagem mais simples, isso significa que o valor de uma parte de software, para qualquer aumento de usuários, aumenta à medida que mais pessoas usam o software em suas máquinas e em seus contextos específicos. Compatibilidade no sentido padrão de uma bem rede é uma das razões por que isso acontece. Assim como é mais valioso para mim ter um aparelho de fax se muitas outras pessoas também têm máquinas de fax, à medida que mais computadores no mundo rodam um sistema operacional ou aplicativo se torna mais fácil comunicar e compartilhar arquivos entre esses computadores. Cada torna-se ligeiramente mais valioso para os usuários existentes quando cada novo usuário entra em cena. (WEBER, 2010. p. 153-154)[27]

Imbuídos dessa filosofia de rede de usuários que colaboram entre si, e com o surgimento da licença GPL, a lógica da formação de um público consumidor/usuário passivo e dependente se inverte, transferindo aos usuários o controle do código-fonte, livre acesso a documentação e a libertação das correntes que os prendiam a fornecedores. Em outras palavras, o poder volta para o consumidor que, aqui, quase nunca é um mero consumidor, mas sim um usuário que, ao consumir o produto também o altera, o enriquece, justamente pela natureza do *software* livre. Com acesso ao código fonte, os usuários (que através do aprendizado poderá se transformar num produtor) podem escolher livremente quais *softwares* atendam às suas necessidades, copiá-los livremente, adaptá-los para outras tarefas e solucionar o problema de integração com outros sistemas.

Dessa forma, os usuários se tornam independente de qualquer fornecedor de TI porque se podem reconhecer as próprias falhas de segurança, corrigir os erros, escrever a documentação ou contratar um terceiro para fazer qualquer uma dessas coisas para você. O conceito de anti-rival foi pensado para o *software* livre. Assim fazendo, é nossa interpretação que o *software* livre confronta a lógica da produção da tradicional indústria de *software* ao retirar o foco da propriedade e do contrato de compra e venda de um bem, e pôr no foco a prestação de serviço.

Com o passar do tempo, e com a convivência, e a concorrência, com o modo não mercantil dos "negócios” dos desenvolvedores de *software* livre, muitas empresas de computação

[27]No original: Software in many circumstances is more than simply nonrival. Operating systems like Linux in particular, and ost software in general, actually are subject to positive network externalities. Call it a network good, or an antirival good (an awkward, but nicely descriptive term). In simpler language, it means that the value of a piece of software to any user increases as more people use the software on their machines and in their particular settings. Compatibilityin the standard sense of a network good is one reason why this is so. Just as it is more valuable for me to have a fax machine if lots of other people also have fax machines, as more computers in the world run a particular operating system or application it becomes easier to communicate and share files across those computers. Each becomes slightly more valuable to existing users as each new user enters the picture.

e informática vão passando a oferecer gratuitamente os *softwares* e a cobrarem por diversas formas de serviços. Entre os modelos de negócios predominantes em empresas capitalistas tradicionais, estão a distribuição de *software* na internet ou em mídia (como DVD) e venda de um conjunto de serviços de suporte técnico e personalização para os usuários. Há outro modelo em que empresas que se especializam em prestar serviços de suporte técnico de vários e diferentes *softwares* em diferentes tipos de *hardware*. Muitas vezes existe uma forte concorrência de preços em torno da distribuição[28] em si (porque qualquer um pode copiá-la) e o suporte técnico é intensivo em trabalho e conhecimento. Há também, empresa fabricantes de *hardware* que encontraram uma forma de incrementar seu faturamento vendendo seus produtos com *softwares* livres e s*oftwares open source*. As empresas desenvolvem *drivers* de *software*, compiladores, aplicativos e até mesmo sistemas operacionais para que os clientes, ao comprar esse *hardware* (um servidor, por exemplo), consigam fazer uso amplo e eficaz da máquina. A utilização do código aberto produz programas de melhor qualidade a um custo muito menor (por que absorve uma enorme quantidade de trabalho voluntário não pago) permite que as funcionalidades prometidas pelo *hardware* sejam exploradas no limite. Então o próprio *hardware* se torna mais valioso (maior valor agregado) e o mercado consumidor para esse produto deve se expandir resultando num aumento de renda para empresa e, por vezes, o domínio de um nicho de mercado (WEBER 2010).

Como estamos vendo, as empresas encontraram diversas maneiras para se beneficiarem comercializando *softwares* livres e de código aberto. Talvez o que seja a forma mais corriqueira de empresas tradicionais de *software* fazerem negócio é vender projetos de *software*. Neste modelo, a empresa vende o trabalho de programação como um serviço e não como um único *software*. Esse tipo de estratégia de vender projetos não é muito diferente de um serviço de táxi, o qual quanto mais carros em operação maior a renda do empresário. Por exemplo, grandes empresas de TI como IBM seguem esse modelo de negócios de venda de projeto quando comercializam serviços "integração de sistemas" (VÄLIMÄKI, 2005). A integração de sistemas é

[28]Aqui cabe uma explicação do emprego da palavra distribuição. Uma distribuição é um versão do sistema operacional Gnu/Linux. Podem ser oriundas de projetos comunitários ou projetos empresariais. Exemplos de distribuições Gnu/Linux comunitárias são o sistema operacional Debian e o Linux Mint. Exemplos de distribuições Gnu/Linux empresariais são Fedora e Ubuntu, as quais são mantidas pelas empresas RedHat e Canonical, respectivamente. O surgimento de distribuições está ligado ao espírito inovador e aberto do *software* livre. Em tese, qualquer pessoa com conhecimento pode criar uma distribuição do sistema operacional Gnu/Linux. Algumas distribuições têm finalidades diferentes como, por exemplo, de ser mais amigável ao usuário, de ser uma distribuição mais enxuta para “rodar” em computadores antigos, de ser um distribuição específica para pessoas com necessidades especiais.

uma adaptação de vários *softwares* distintos para que se comunique entre eles trazendo ganhos de tempo e eficiências para clientes. Esse tipo de negócios é relativamente comum em grandes corporações que precisam integrar a comunicação e processos de diversos setores, processo esse imensamente facilitado se os *softwares* são livres.

Há também o SaaS (*Software as a Service*)[29] serviço de *software* na nuvem, ou seja, na internet. Usando nuvem como ambiente de instalação, uso e canal de distribuição. O C*loud Computing* inaugurou uma era de várias novas maneiras de fazer negócios de *software*. Algumas empresas disponibilizam o código-fonte na nuvem sob uma licença aberta ou livre para acesso universal. Há empresas que trabalham com versão comunitária do *software* e versão profissional ou empresarial (chamada de *enterprise*), a qual precisa de uma assinatura mensal (como um plano de TV a cabo) para acessar[30]. Na maioria dos casos, a necessidade da assinatura é justificada como uma maneira de remunerar programadores, pagar infraestrutura (como hospedagem do serviço em *data center* de ponta e espaço em disco para armazenagem de dados do cliente). Há outras empresas que trabalham dentro da lógica do SaaS, porém o pagamento de uma assinatura é necessário quando o usuário tem interesse em aplicativos ou modelos especiais.[31] Contratar a assinatura de um *software* pode ser visto como uma combinação dos dois modelos tradicionais: pagamento para ter acesso e uso e acesso ao código-fonte da maior parte do *software*. Assinatura é uma forma mais interativa para vender *software* como um produto on-line com os serviços *add-on*[32] sob medida para o cliente. Na prática, qualquer empresa de *software* de venda de serviços baseados na nuvem podem ser classificados seguindo o modelo de assinatura (VÄLIMÄKI, 2005).

Por último, diferentes modelos de negócios de *software* de *commodities* têm surgido. Aqui, um produto principal ou de um componente padronizado está disponível gratuitamente ou com preços baixos. “Por definição, todas as vendas no modelo de *commodities* são baseadas em

[29]*Software* como Serviço, em português.

[30]Um exemplo desse tipo de estratégia comercial é da empresa SugarCRM. A empresa possui uma versão comunitária com licença GPL Affero. Contudo, a empresa oferece aos clientes uma versão enterprise com vários recursos, a qual precisa de uma assinatura.

[31]Exemplos de organizações que adotam esse modelo de negócios são a Drupal e WordPress

[32]Em computação o termo Add-ons refere-se a módulos de hardware ou software (sub-sistemas ou pseudo-programas) que suplementam ou aumentam as ferramentas e possibilidades de uso ou características originais onde são utilizados. Também são utilizados outros termos tais como: plugins, extensões, snap-ins e vários outros especificados de acordo com o fabricante.

meios indiretos que aproveitam a base de usuários potencialmente grande e dinâmica. Por exemplo, os serviços de add-on, produtos integrados e marcas são fontes essenciais de receitas indiretas" (VÄLIMÄKI, 2005, p. 21). Ainda Segundo Välimäki (2005, p. 21) "[...] o *software* de código aberto podem ser combinados em qualquer popular modelo de negócio de software. Não faz sentido falar de negócio de *software* de fonte aberta como se isso fosse denotar a um modelo de negócio de *software* específico". Mikko Välimäki tem razão em parte. Quando se refere a negócio, ou seja, vendas de serviços tais como customização, consultoria, manutenção, treinamento e desenvolvimento não há muitas diferenças. Contudo, não se pode esquecer que o código-fonte desses programas é aberto, a licença permite que usuários possam estudar adaptar e compartilhar esses programas irrestritamente. Dito de outra forma, teoricamente é possível que apareça um novo concorrente em qualquer parte do mundo. Algo assim, raramente ocorre com as grandes empresas de *software* proprietário. Então, afirmar que não há um modelo de negócios diferente é exagero.

Não se pode comparar a natureza dos *softwares* livres/abertos com a do fechado, pois a forma de monetizar é diferente. Uma das mais significativas fontes de renda que as empresas de *software* proprietários possuem é o pagamento de *royalties*, entretanto essa forma de renda é muito menor para empresas fora do modelo do *software* proprietário.[33] Os próprios projetos buscam formas de se financiar. Sejam organizando eventos, pedindo doações de pessoas físicas e jurídicas, através de financiamento coletivo ou desenvolvendo uma "lojinha" com camisetas, adesivos, casacos, mochilas com a logo do projeto (que é um *software*). Muitas comunidades se mantêm dessa maneira uma vez que não possuem fins lucrativos. O resultado é essa que difusão voluntária do *software* livre/aberto ajuda a aumentar a fatia de mercado de empresas que prestam serviços com esses mesmos *softwares*.

A apropriação privada do trabalho coletivo pode representar um risco a motivação dos desenvolvedores. Ao saber que uma empresa promove a apropriação privada de seu *software* visando unicamente o lucro, ainda que o mantenha livre, alguns membros dessa comunidade poderão abandonar o projeto, por se sentirem explorados e criarem um projeto alternativo completamente comunitário. Outra situação é a empresa, para ganhar mercado consumidor,

[33]Vender uma licença de uso de um *software* livre não é proibido. Porém, essa prática não é comum, pois geralmente as pessoas conseguem baixar da internet sem nenhum tipo de custo um programa livre. Mas uma empresa pode ter feito mudanças significativas num *software* e, mesmo o *software* sendo livre, a empresa pode cobrar de seus clientes um preço simbólico por cada cópia desse *software* que for instalado. Assim, a empresa consegue auferir uma receita com vendas de licenças.

contratar as pessoas da comunidade para seus quadros produzindo o seu enfraquecimento enquanto movimento social. Assim, os *softwares* que podem trazer mais lucratividade poderão ganhar preferência pondo de lado *softwares* sem apelo comercial, como software para inclusão de pessoas com problemas de visão. A garantia de continuidade do *software* livre não passa apenas pelo desenvolvimento de *software*, mas também pela motivação das pessoas envolvidas no movimento.

4.2 A MOTIVAÇÃO DO TRABALHO COLABORATIVO

A quantidade de comunidades de *software* livre ativas em todo mundo é desconhecida, mas calcula-se que exista, pelo menos, uma para cada distribuição do sistema operacional GNU/Linux, para cada *software* (jogos, editores de texto, edição e editoração de imagens), além dos grupos de usuários, listas de ajuda e sites de projetos (FISCH, 2008). A diversidade desse ecossistema inviabiliza uma pesquisa que apresente dados precisos, além do mais, é comum os seus integrantes flutuarem de um projeto para outro e trabalharem em mais de um projeto simultaneamente. Por exemplo, em um projeto é tradutor voluntário, em outro trabalha auxiliando novos usuários tirando dúvidas em fóruns e em outro atua como um desenvolvedor. Também o conceito de desenvolvedor precisa ser expandido, pois o público entrevistado pela pesquisadora Patrícia Fisch se considera desenvolvedor de *Software* Livre não apenas quando desenvolvem, corrigem ou aperfeiçoam o código-fonte, mas também quanto redige a documentação ou quando promovem a disseminação da filosofia do projeto ou conhecimento. As respostas indicam, também, que os participantes sentem-se engajados dentro de suas práticas na comunidade, não fazendo distinções significativas entre os resultados das práticas que exercem. As comunidades possuem uma anarquia autogerida. Nota-se que não há uma preocupação com a eficiência (aqui, a palavra eficiência deve ser compreendida no contexto da eficiência econômica) do projeto que estão desenvolvendo, pois não é construído para levar ao mercado.

A pesquisa aponta que é relativamente comum encontrarmos, entre os membros das comunidades, pessoas que trabalham comercialmente com os mesmos *softwares*. Dito em outras palavras, muitos trabalhadores em empresas privadas e públicas integram comunidades de *software* livre para acompanhamento e colaboração espontânea, contudo apenas nas comunidades correlatas às suas atividades profissionais (FISCH, 2008). Lakhani e Wolf, em seus estudos, questionaram seus entrevistados se recebem alguma renda para trabalhar em algum projeto, 87%

responderam negativamente, contudo 55% afirma que contribui com código durante seu horário de trabalho (alguns escondidos de seu superior). “A combinação dos que receberam compensação financeira direta e aqueles cujos supervisores sabiam do seu trabalho no projeto criou uma categoria de 'contribuidores pagos', constituídas por aproximadamente 40% da amostra.” (LAKHANI; WOLF, 2005).

O trabalha voluntário permanece como sendo a principal forma de contribuição, contudo há também o trabalho remunerado geralmente realizado por empresas interessadas no produto que o projeto poderá gerar. Imaginemos uma grande comunidade como a Apache. Há muitos trabalhadores voluntários, mas há aqueles remunerados. Ambos constituem a comunidade do projeto Apache. O esforço dos membros do projeto pode ser medido em número de horas semanais gastas com assuntos relacionados ao projeto. Esta medida tem sido utilizada em estudos anteriores e fornece uma métrica da contribuição do participante e o tamanho de seu interesse em projetos (LAKHANI e WOLF, 2005). Em sua pesquisa, Lakhani e Wolf, perguntaram aos entrevistados quantas horas na última semana foram alocadas em cada um dos projetos de que faz parte. Os entrevistados responderam que tinham ocupado, em média, 14,1 horas de sua semana em todos os seus projetos e 7,5 horas sobre o projeto de maior interesse. No geral, os contribuidores pagos ocupam mais de dois dias de trabalho por semana e contribuidores voluntários estão gastando mais do que um dia por semana em projetos. O subsídio financeiro implícito nos projetos é substancial. Em 2001, o *Bureau of Labor Statistics*, dos Estados Unidos, informou que a remuneração, por hora, foi em média de US$ 30,23 dólares para os programadores de computador. Assim, através de um cálculo simples, constata-se que a contribuição financeira média semanal para projetos de *software* livre/*open source* é US$ 353,69 de trabalho voluntário e US$ 535,07 dólares dos contribuidores pagos via seus empregadores.

Isso nos faz entender que, na prática, as comunidades tanto são um local de encontro e trabalho de ativistas do Movimento Software Livre quanto um apêndice das empresas de TI. A etapa de desenvolvimento mais recente do capitalismo nos coloca diante de novas organizações do trabalho e suas formas de exploração. O *software* livre representa uma ruptura com a tradicional indústria de TI, mas a busca por um modelo comercialmente escalável para ele condiciona os ativistas num cenário em que é necessário adequar a subversividade libertadora do *copyleft* ao hermético mercado de TI. Apesar desse dilema que muitos ativistas se deparam, muitos são forçados a pôr de lado suas convicções ideológicas para conseguir trabalhar e se sustentar. Mesmo assim, a principal motivação apontada é a contribuição de caráter social. Diz

FISCH:

> A motivação pela contribuição de caráter social é considerada muito importante por 88% dos entrevistados, e a maior motivação em todo o grupo de entrevistados. A disseminação de conhecimentos para a sociedade está em segundo lugar com 60% das respostas e o reconhecimento com 40%. Neste grau de importância o que menos é levado em consideração em relação ao trabalho executado é a motivação financeira, tendo 8% de respostas. Percebe-se o grande interesse pela contribuição que possa ter um retorno social para a sociedade em detrimento do ganho financeiro. Isto se deve ao fato de que as comunidades de software livre produzem com a premissa de que tudo o que é distribuído deve retornar para a sociedade por se tratar de bem imaterial, que não se desgasta pelo uso. (FISCH, 2008, p.66)

Os dados são esclarecedores ao que se refere a motivações. Nas pesquisas[34] elaborou-se um estudo mais profundo e buscando identificar motivações pessoais, sociais e de trabalho em muitos projetos diferentes. Quando perguntado qual a principal motivação para fazer parte do projeto, 58% dos entrevistados apontaram como a principal razão para a contribuição o auxilio ao usuário, essa resposta aparece tanto para os trabalhadores remunerados pelos projetos quanto os trabalhadores voluntários. Porém as motivações em escrever código para o Projeto é intelectualmente estimulante foi citado por 44,9% dos entrevistados. Melhorar as habilidades de programação, uma motivação relacionada à melhoria do capital humano aparece com 41,8% dos participantes dizendo que era um motivador importante. Aqui, podemos observar a heterogeneidade das motivações. Ao mesmo tempo em que se têm muitas respostas positivas para uma demanda social, auxiliar os usuários desse *software*, tem-se também o lado mais individualizado em que buscam aprimorar conhecimento e testar suas capacidades. Cerca de um terço da amostra indicou que a crença de que "o código fonte deve ser aberto” e que isso é uma obrigação comunitária foi apontada como uma importante razão para a sua participação. Essa resposta foi seguida de perto por aqueles que indicaram que contribuem por que se sentem na obrigação de dar algo de volta à comunidade *software* livre com 28,6% de respostas.

Outra questão central na pesquisa foram as motivações dos desenvolvedores para contribuir com a criação de um “bem público”. Os indivíduos contribuem, em média, com 14 horas de trabalho por semana. É um importante indicador de esforço e dedicação. Umas das questões mais importantes e pouco estudada na pesquisa de Lakhani e Wolf (2005) são os aspectos econômicos das comunidades e dos projetos. A presença de participantes pagos, 40% da

[34]As duas pesquisas utilizadas nesse trabalho foram aplicadas em locais geograficamente distintos. As entrevistas conduzidas por Fisch estão circunscritas ao Brasil, especificamente, ao estado do Paraná. Já as pesquisas de Lakhani e Wolf foram respondidas por pessoas que moram na Europa Ocidental, Estados Unidos e Canadá.

amostra do estudo, indica que há uma apropriação de conhecimento coletivo por parte das empresas de TI e mais do que isso, há uma apropriação do trabalho voluntário alheio. Observando, cuidadosamente esse cenário, constatamos, contudo, que as empresas pagam desenvolvedores para atuarem no seio de projetos de *software* livre, nos quais a propriedade intelectual permanecerá (ou se tornará) livre. Ou seja, as empresas privadas de tecnologia também estão se tornando uma importante fonte de recursos para os projetos de *software* livre e *open source*. É uma via de duas mãos. “A contribuição das empresas para a criação de um bem público levanta questões sobre incentivos à inovação com potenciais concorrentes. Além disso, a interação entre os participantes pagos e voluntários dentro de um projeto levanta questões sobre os limites das políticas de colaboração firmes e apropriadas.” (LAKHANI e WOLF, 2005). Esse tema é fundamental para compreender como se organiza esse trabalho voluntário/remunerado que esculpiu o “moderno” trabalho colaborativo. Como as empresas privadas, os projetos, as comunidades, as organizações civis (como Ongs, fundações e associações) e o Estado se organizam para apoiar a produção colaborativa se apropriando do conteúdo intelectual ao mesmo tempo em que buscam assegurar que ele permaneça livre.

4.3 TRABALHO COLABORATIVO E PRODUÇÃO COLABORATIVA EM REDE

O trabalho colaborativo está tão associado à produção colaborativa que fica quase impossível dissociar um do outro. Yochai Benkler em seu livro “*The Wealth of Networks”* (A Riqueza das Redes) ao analisar o fenômeno do *software* livre o descreve da seguinte maneira:

> Ele sugere que o ambiente de rede torna possível uma nova modalidade de organização da produção: radicalmente descentralizada, colaborativa e não proprietária; com base na partilha de recursos e resultados amplamente distribuídos, indivíduos frouxamente ligados que cooperam uns com os outros, sem depender de quaisquer sinais do mercado ou comandos gerenciais. Isto é o que eu chamo de "produção de bens comuns baseada em pares." (BENKLER, 2006 p.60)[35]

Para chegar nesse conceito, Benkler, utiliza as teorias de produção da economia industrial, em especial, as teorias de Ronald Coase, que define dois modos de interação que os indivíduos respondem aos incentivos de produção. Num deles, o indivíduo reage diretamente às demandas do mercado, ao passo que no outro, responde a uma cadeia hierárquica de empresas. BENKLER

[35]No original: It suggests that the networked environment makes possible a new modality of organizing production: radically decentralized, collaborative, and nonproprietary; based on sharing resources and outputs among widely distributed, loosely connected individuals who cooperate with each other without relying on either market signals or managerial commands. This is what I call “commons-based peer production.

(2006), a partir desses estudos, apresentou uma nova modalidade de modelo de produção, a qual denominou de produção de bens comuns baseada em pares. O modo de produção do FOSS (*Free and Open Source Software*) possui essa característica inovadora de produção. Discutimos o formato de desenvolvimento do segundo capítulo, especialmente, quando falamos do modelo de desenvolvimento catedral e modelo de desenvolvimento bazar. Benkler (2006) absorve essas definições desenvolvidas pelo Eric Raymond e busca formalizar dentro de teorias econômicas esperando, assim, montar uma explicação científica para o fenômeno. O mérito de Benkler (2006) foi a formalização desse processo inovador de produção em que há colaboração de pessoas geograficamente dispersas, sem vínculos significativos, com diferentes motivações (conforme vimos anteriormente), experiências profissionais e culturais distintas tendem a gerar, sob certas circunstâncias, um melhor sistema de produção de informação e bens culturais do que as formas de produção via mercado. Para o Benkler (2002 p. 9), "As vantagens da produção entre pares são uma melhor identificação e alocação da criatividade humana. Estas vantagens parecem ter se tornado salientes, porque a própria criatividade humana tornou-se saliente."[36] Tal modo de organização da produção, alicerçado na colaboração dos pares para a produção de bens comuns, é fundamental para o desenvolvimento crescente destas comunidades. Elas garantem uma menor perda de informações graças à menor burocracia envolvida e da escolha voluntária das tarefas de cada colaborador, um maior nível de criatividade devido à diversidade dos colaboradores e um alto grau de eficiência gerado pelo caráter de modularidade e granularidade dos projetos.

Esse modelo de produção de bens comuns baseada em pares é uma forma de produção fora do mercado, até agora. Os projetos de software livre desde a publicação do Manifesto GNU foram pensados como projetos comunitários com trabalho voluntário. O próprio desenvolvimento do *kernel* do GNU/Linux foi um trabalho individual desenvolvido por Torvalds sem a pretensão de se tornar uma mercadoria. Porém, Benkler, faz uma interessante observação, em seu Livro "*The Wealth of Networks"*. Diz Benkler:

> Não há fabricantes de automóveis não comerciais. Não há fundições de aço voluntário. Você nunca iria optar por ter a sua principal fonte de pão depende de contribuições voluntárias dos outros. No entanto, os cientistas que trabalham em institutos de pesquisa não comerciais financiados por instituições educacionais sem fins lucrativos e os subsídios do governo produzem a maior parte da nossa ciência básica. Redes de cooperação generalizados de voluntários escreveram o software e as normas de funcionamento da maior parte da Internet e o que permitem que fazemos com ela. Muitas pessoas recorrem à National Public Radio ou a BBC como uma fonte confiável de notícias. O que, sobre a informação, que explica essa diferença? Por que dependemos quase exclusivamente de mercados e empresas comerciais para produzir carros, aço e trigo, mas muito menos para as informações mais críticas que as sociedades avançadas dependem? Será esta

[36] No original: The advantages of peer production are, then, improved identification and allocation of human creativity. These advantages appear to have become salient, because human creativity itself has become salient.

> uma contingência histórica ou há algo sobre a informação como objeto de produção que torna a produção não mercantil atraente? (BENKLER, 2006 p.35)[37]

Há na produção do conhecimento, da informação e de bens culturais algo em comum. Todos são bens intangíveis e com um processo de produção diferente daquele da indústria fabril e da agricultura. Ainda no século XX a produção e a distribuição de bens imateriais era intimamente ligada ao mundo material. Na nascente indústria de computadores, quando os fabricantes de *hardware* forneciam o *software*, a propriedade dos meios de produção ficava com o proprietário das empresas e, portanto, o código-fonte foi livremente disponível. Os programas eram um componente do *hardware*. A imaterialidade não havia sido percebida como uma oportunidade de negócio. Assim, música não existia sem o disco, a literatura era impossível sem o livro de papel, o filme fora do cinema só poderia ser vistos utilizando fitas K7 e o *software* não era percebido sem o *hardware*. No entanto, conforme discutimos no primeiro capítulo, com o avanço da tecnologia, em especial o desenvolvimento da indústria do *software* e da internet, bens intangíveis se tornaram independentes de suas contrapartidas físicas. Foram convertidos em bits. Contudo, o processo de exploração do trabalho era similar a aquele das fábricas, por exemplo. Uma vez que os *software* e o *hardware* se divorciaram, "houve novos jogadores: os capitalistas de *software*, que precisavam de uma maneira de manter o controle similar do *software*. Fechar o código-fonte foi a maneira mais simples de fazê-lo" (CHOPRA; DEXTER).

O usuário é incapaz de perceber a infraestrutura do produto ou mudá-lo para atender às suas necessidades específicas. O modelo FOSS, permite que tanto os usuários quanto trabalhadores da TI possam modificar o programa. É uma mudança importante para a visão clássica do mundo da manufatura (CHOPRA e DEXTER). O *software* livre é a oportunidade de um reencontro do homem com os frutos do seu trabalho, "consigo mesmo e com os outros homens que se relacionam com ele nesse processo de produção [...]" Porém, não podemos cair na tentação de afirmar que a informação é a principal fonte de valor da sociedade pós-industrial, na qual, segundo teóricos da futurologia, já estamos vivendo (SÖDERBERG, 2002). A informação é

[37]No original: "There are no noncommercial automobile manufacturers. There are no volunteer steel foundries. You would never choose to have your primary source of bread depend on voluntary contributions from others. Nevertheless, scientists working at noncommercial research institutes funded by nonprofit educational institutions and government grants produce most of our basic science. Widespread cooperative networks of volunteers write the software and standards that run most of the Internet and enable what we do with it. Many people turn to National Public Radio or the BBC as a reliable source of news. What is it about information that explains this difference? Why do we rely almost exclusively on markets and commercial firms to produce cars, steel, and wheat, but much less so for the most critical information our advanced societies depend on? Is this a historical contingency, or is there something about information as an object of production that makes nonmarket production attractive?"

o resultado de trabalho humano, pois ela era necessária para se construir um machado de sílex. A mudança está em que a informação foi mercantilizada. Semelhantemente a outros recursos, “a informação é reivindicada pela expansão capitalista a ser produzida pelo trabalho assalariado e para dentro de um mercado” (SÖDERBERG, 2002 p.4).[38]

Sendo assim, a informação – podemos pensar no conhecimento para ampliar o raciocínio - livre contribui para a disseminação de uma inovação. Se um fornecedor de uma distribuição GNU/Linux desenvolve uma inovação que se torna popular, os fornecedores de outras distribuições GNU/Linux também vão adotar, porque o código-fonte do *software* é livre. Esse modelo ganha-ganha é um grande trunfo do *software* livre tanto para desenvolvedores quanto para usuários. Johan Söderberg (2002) comenta em seu artigo, *Copyleft vs. Copyright: A Marxist Critique (Copyleft vs. Copyright: Uma Crítica Marxista)* que um dos argumentos defendidos por teóricos do que ele chama, a economia política do *software* livre é que o próprio mercado vai pressionar a indústria de *software* privativo para colocar suas propriedades intelectuais sob uma licença livre. A explicação de como isso poderá ocorrer é semelhante ao que ocorre com a língua, em que se torna mais importante e útil na medida em que mais pessoas falam e escrevem. A ideia de que todos os consumidores de *software* que foram excluídos do mercado por não querer (ou não conseguir) pagar por uma licença de uso de um programa, agora, usarão um *software* que não cobra pela licença de uso. Esse cenário poderá deslocar os usuários das empresas de *software* proprietários para empresas de *software* livre. Por exemplo, se a Microsoft – que é a monopolista no mercado de sistemas operacionais – começa a sofrer uma grande concorrência de empresas que oferecem um sistema operacional livre (isto é, de graça e com o código-fonte aberto) e percebe que seus clientes estão abandonando o seu produto (nesse caso, o Microsoft Windows) possivelmente a empresa começa a cogitar a possibilidade de distribuir o seu sistema operacional sem custo e com o código-fonte aberto.[39]

A histórica plasticidade do capitalismo, nos leva a crer, contudo, que não existem soluções mágicas via mercado. O que muitas empresas de *software* proprietário procuram é reduzir seus

[38]No original: […] information is claimed by capitalist expansion to be produced by wage labour for and within a market.

[39]Ao que parece, esse tipo de situação descrita no parágrafo encontra lastro com a realidade. A Microsoft anunciou no dia 12 de novembro de 2014, em seu site, que estava criando a.NET Fundation (lê-se dot net) voltada para a comunidade open source. Para sinalizar a seriedade desse comunicado, a Microsoft colocou parte do framework.Net para essa fundação gerenciar abrindo o código-fonte das soluções para servidores. No entanto, a principal intenção da Microsoft é desenvolver uma comunidade que a ajude a co-desenvolver seus softwares comunicando e consertando erros. Assim, podem reduzir os custos de trabalho contratando menos programadores para encontrar e corrigir erros.

custos e ampliar a apropriação privada do coletivo. Ora, se uma empresa tem uma grande comunidade que trabalha em sintonia comunicando e consertando erros, sugerindo melhorias e difundindo os *softwares* em suas redes de relacionamentos não são mais necessários tantos trabalhadores formais nos quadros da empresa uma vez que a empresa terá uma porção de serviços intelectuais altamente especializados gratuitos. É claro, dirão alguns, que em troca há o acesso ao conhecimento e a gratuidade para usar os *softwares*. Mesmo assim, a apropriação promovida pela empresa supera a apropriação promovida pelos usuários. Segundo Söderberg (2002):

> Certamente, então, há interesses contraditórios entre os dois lados, não provenientes tanto da atividade formal (software livre ou fechado), mas a partir de seus agentes (comunais ou comerciais). Se as comunidades se tornarem produtores diretos de valor para o capital, deve-se esperar surgir uma relação antagonista semelhante a que existe entre capital e trabalho. (Söderberg 2002 p. 10 tradução nossa)[40]

À medida que os resultados produtivos dos projetos comunitários se tornam uma fonte de renda para empresas e, além disso, tornam-se um "modelo de negócios" em que cada empresa quer ter uma comunidade em torno de seu projeto para dar suporte podemos afirmar que o "O trabalho social está fazendo incursões dentro da própria produção capitalista, que precisa utilizar a capacidade cooperativa e comunicativa da força de trabalho, a fim de manter-se competitiva" (SÖDERBERG 2002 p.11)[41]. O que estamos assistindo é uma alteração nas relações de produção? A drástica mudança imposta pelo *software* livre – principalmente pela criação da cultura do *copyleft* -, "[...] que cresceu rapidamente de forma espontânea e totalmente fora das estruturas de produção mercantis, construiu uma economia paralela que supera a economia de mercado" (SÖDERBERG 2002 p.11).[42]

Esse cenário pode ser entendido como uma indicação de como as forças produtivas estão desconstruindo as relações estabelecidas de produção. O conhecimento, que o capital possui, protegido através da propriedade intelectual, é muitas vezes, apropriado de comunidades, em primeiro lugar, se é *software* feito por *hackers* ou se são conhecimentos tradicionais passado de geração em gerações de agricultores. A estratégia de combate à apropriação privada do conhecimento humano – intelecto geral - seria reconhecer o direitos de propriedade de direitos

[40]No original: Certainly then, there are contradictory interests between the two sides, not originating so much from the formal activity (free or closed software), but from its agents (communal or commercial). If communities become direct producers of value for capital, an antagonist relationship similar to the one between capital and labour should be expected to emerge.

[41]No original: Social labour is making inroads within capitalist production itself, which needs to utilize the cooperative and communicative capacity of the workforce in order to stay competitive.

[42]No original: […] spontaneously and entirely outside of previous capital structures of production. It has built a parallel economy that outperforms the market economy.

autoral comunitário. Esta tática é, essencialmente, o caminho construído pela *Free Software Fundation* e o *Copyleft*. SÖDERBERG (2002) procura explicar que a superação de um modelo pelo outro não passa pelo tipo de licenciamento, ao contrário passa pelo o que a licença preserva:

> A linha não é desenhada entre a propriedade e as licenças, mas entre formas opostas de licenças, uma apoio a um regime de propriedade e a outra um comunal. Quem vai prevalecer? Recordando o materialismo histórico, um de sua fundação afirma que "a classe que governa através de um período, ou emerge triunfante após um conflito de época, é a classe mais adequada, mais capaz e disposta, para presidir o desenvolvimento das forças produtivas no dado tempo" (SÖDERBERG 2002 p. 11. tradução nossa).[43]

A ideia de luta de classes na qual Söderberg se refere é entre os que defendem *copyright* e os que defendem *copyleft*, já que ambos representam pensamentos distintos de grupos distintos. *Software* é trabalho, diversão, aprendizado, novas amizades e debates. Sem restrições geográficas, os usuários colaboram uns com os outros sem a mediação direta de dinheiro ou política. “Sem se preocupar com direito autoral, dão e recebem informação sem pensar em pagamento. Na a ausência do Estado e dos mercados para mediar os laços sociais, as comunidades de rede são formadas pelas obrigações mútuas criadas por dádivas de tempo e ideias” (BARBROOK 1998, p.5)[44].

Os projetos comunitários liderados pelos entusiastas do software livre estão expostos ao assédio do grande capital, contudo notamos a dificuldade do capital dominar esse movimento, apesar de grupo open ser mais amigável a ideia de competir no mercado com as empresas de software proprietário. Apesar de existir diferenças entre o grupo free e open, ambos são aliados. A indústria do software, que antes combatia, passou ao tolerar, para, agora absorver algumas práticas e procedimentos característicos do software livre. Apesar das licenças utilizadas não serem compatíveis com o *copyleft*, é notório que uma mais uma pequena batalha foi ganha. Se o grande capital da indústria do software se rende ao modelo do software livre/aberto, o mesmo capital tenta se beneficiar dos projetos e do conhecimento gerado por eles. Porém, dessa vez, toda propriedade intelectual é livre e tudo o que for produzido deverá ser compartilhado.

---

[43]The line is not drawn between property and licenses, but between opposing forms of licenses, one supporting a proprietary regime and the other a communal. Who will prevail? Recalling historical materialism, one of its foundation states that "the class which rules through a period, or emerges triumphant after epochal conflict, is the class best suited, most able, and disposed, to preside over the development of the productive forces at the given time"

[44]No original: Unconcerned about copyright, they give and receive information without thought of payment. In the absence of states or markets to mediate social bonds, network communities are instead formed through the mutual obligations created by gifts of time and ideas.

5. CONCLUSÃO

Esse trabalho foi dedicado a compreender o fenômeno sócio-político-econômico do *Software* Livre, seu surgimento, sua ostentação e transformações internas. Do ponto de vista econômico, viu-se que, as quatro bandeiras das liberdades fundamentais do usuário em ter acesso ao código-fonte dos *softwares*, de os estudar, os modificar e os distribuir as modificações feitas deram origem a um modelo de negócios – o modelo bazar. Do ponto de vista político, deu origem ao Movimento Software Livre (MSL) cujos objetivos eram mais amplos, pois visavam romper com o *establishment* da indústria da TI e sua lógica de mercantilização de um mecanismo jurídico que garantia que um *software* criado livre permanecesse livre: a licença batizada de *General Public Licence* (GPL). Essa licença é ela mesma uma inovação jurídico-política ao mesmo tempo fruto e indutora de uma transformação sócio-econômica; sem ela, a tarefa de copiar códigos-fontes abertos e construir novos *softwares* fechados (a serem mercantilizados) seria muito mais simples. A GPL possui uma característica importante de produzir um "viral" e isso favorece a criação de novos softwares livres. Estas características compõem o que foi a cultura do *copyleft*.

Contudo, divergências políticas, técnicas e econômicas marcaram a ruptura sofrida pelo Movimento S*oftware* Livre. Politicamente, o que pesava era o discurso à esquerda do Stallman, que era a principal personalidade do movimento e pessoalizava, canalizando para a *Free Software Fundation* (FSF), toda a "glória" recebida pelas iniciativas do movimento. As divergências técnicas ficam claras no livro "A Catedral e o Bazar", no qual Eric Raymond compara o método de desenvolvimento das tradicionais empresas de *software* ao método de construção das grandes catedrais medievais. Ele inclui a FSF entre as organizações catedral, ou seja, com uma hierarquia vertical, lenta e burocrática. Raymond compara o método de desenvolvimento do Linux com um bazar, ou seja, com uma hierarquia horizontal, descentralizado, ágil e sem burocracias. Em outras palavras, a FSF é similar a qualquer empresa de software tradicional.

O fator econômico está conectado a uma aparente hostilidade da *Free Software Fundation* ao mercado tradicional de TI. Para Stallman, não é possível fazer concessões ao *software* proprietário, no entanto para o grupo, *open source*, para quem concessões viabilizaram tanto a melhoria técnica dos *softwares*, como sobrevivência física dos seus desenvolvedores e mesmo a sobrevivência política do movimento, redefinindo novas bases, menos políticas-ideológicas. Esse racha ocorrido no Movimento Software Livre deu origem a dois

movimentos autônomos que fazem a defesa do código-fonte: O Movimento *Software* Livre e o Movimento *Open Source*.

No geral, o desenvolvimento descentralizado, horizontal e cooperativo é responsável pelos *softwares* de alta complexidade como o Gnu/Linux e com uma licença que coletiviza a propriedade intelectual. Essa possibilidade abriu as portas para uma nova discussão sobre os limites e alcances da propriedade intelectual no capitalismo contemporâneo. A possibilidade lançada pela GPL e pelo método de desenvolvimento colaborativo e horizontal do Linux, trouxe à tona o debate sobre a eficiência e relevância econômica e social da propriedade privada. A notória superioridade do trabalho livre e colaborativo no desenvolvimento de softwares alcançou a produção de outras tecnologias e colocou em questão a produção baseada no assalariamento tradicional (tornando sua gestão pela empresa capitalista mais diferenciada e flexível).

No que há de mais específico, analisou-se os modelos de negócios da indústria da TIC tradicional e do *Software* Livre/*Open Source*, pois estudando esses modelos de negócios entendemos as estratégias empresarias que possibilitam a convivência e mesmo a prosperidade de ambos na atualidade. Esta convivência não é, contudo, bem vista por todos, uma vez que é possível perceber que a cultura do *copyleft* está, paulatinamente, aumentando de importância e se tornando a referência dos novos empreendimentos e vendo antigos detratores adotarem suas práticas e seus métodos, embora muito menos por adesão filosófica e muito mais por perceber uma nova possibilidade de renda através da apropriação do trabalho coletivo.

Se aos empresários capitalistas tradicionais, se juntam novos empreendedores oriundos das *startups*, buscando todos estas maneiras de fazer do trabalho coletivo não pago das comunidades uma mercadoria: simultaneamente os militantes do *software* livre têm se engajado em projetos comunitários buscando autonomia das relações de trocas mercantis. Embora muitos desses projetos se tornem empresas bem sucedidas com foco na prestação de serviços em *software* livre/aberto, a produção descentralizada, livre e cooperativa permanece uma característica marcante dos projetos e não há sinais de que essa prática comunitária possa acabar.

Para entendermos como esses projetos funcionam na prática buscamos identificar, através da análise de dois trabalhos de pesquisa de campo, quais são os perfis e as motivações dos integrantes desses projetos. Podemos apurar que muitas empresas adequaram sua estratégia de negócios na direção do *software* livre/aberto sem perder de vista seu viés comunitário que funciona como lastro tanto para o mercado quanto para a própria comunidade. O financiamento de projetos de desenvolvimento através da remuneração de seus integrantes pode ser apontada

como a principal estratégia dessas empresas. A comunidade e a empresa, muitas vezes, unem-se num só organismo. Apesar disso, o trabalho voluntário coletivo permanece sendo o predominante na maioria dos projetos já que muitos desenvolvedores pagos também são voluntários em outras comunidades. A essência filosófica do *software* livre permanece sendo o motor motivacional quando descobrimos que uma das principais motivações para os integrantes desses projetos é o código-fonte ser livre/aberto. Essa informação é fundamental, porque nos confirma que muitos participantes estão nos projetos por ideologia e não para aumentar sua renda ou procurar emprego. Na pesquisa executava por Lakhani e Wolf (2005), revelou que o tempo médio de trabalho voluntário por semana era, em média, 14,1 horas em todos os seus projetos e 7,5 horas sobre o projeto de maior interesse e que os trabalhadores pagos trabalhavam em média dois dias enquanto os voluntários trabalhavam em média um dia por semana.

Como balanço, vê-se que estamos assistindo a transformação na forma de organização do trabalho coletivo, que a história do Movimento *Software* Livre (MSL) é ela própria fruto e estímulo destas transformações, que o Movimento *Open Source* (MOS) tanto pode significar uma perda da radicalidade ideológica do MSL como uma ganho na medida em que abre possibilidades novas de sua sustentação econômica. Os tempos atuais mostram que há formas de convivência entre o trabalho coletivo organizado autonomamente pelas comunidades (redes de programadores/desenvolvedores). Mostram também que pode-se viabilizar algum ganho econômico para produtores sem que subordinem, ou que façam apenas parcialmente, ao assalariamento. Por fim, mostram que se a cultura do *copyright* levou, via estímulo do ganho privado, ao crescimento da riqueza e ao desenvolvimento da tecnologia até nossos dias, daqui em diante a cultura do *copyleft* promete eficiência econômica junto com ganhos sociais, além de satisfação individual via reconhecimento dos pares dentro de suas comunidades.

6. REFERÊNCIAS

BARBROOK, Richard. Hi-Tech Gift Economy. **First Monday**, Cidade, v. 3, n. 12, p. Dec. 1998. Disponível em: <http://firstmonday.org/ojs/index.php/fm/article/view/631/552> Acesso em: 12 nov. 2014

BENKLER, Yochai. Coase's Penguin. **The Yale Law Journal.** New Haven**.** v. 112, n, pp. 371-399, dec. 2002. Disponível em: <http://www.yalelawjournal.org/article/coases-penguin-or-linux-and-the-nature-of-the-firm>. Acesso em: 8 jun. 2015

_____. **The Wealth of Networks:** How Social Production Transforms Markets and Freedom. New Heven: Yale University Press, 2006.

BRETON, Philippe. História da informática. São Paulo: Editora UNESP, 1991.

CAMPBELL-KELLY. Martin et al. **Computer:** a history of the information machine. 3rd ed. Boulder: Westview Press, 2014.

CARMONA, André I. S. **O software livre no limite da propriedade intelectual:** uma breve apresentação. 2008. 43 f. Monografia (Graduação em Ciências Econômicas) – Departamento de Ciências Econômicas - Universidade Federal de Santa Catarina, Florianópolis, 2008.

CASTELLS, Manuel. A era da informação: economia, sociedade e cultura. In: _____. **Sociedade em Rede**. 3. ed. São Paulo: Paz e Terra, 1999. Vol. 1

CHOPRA, Samir. DEXTER, Scott. **The Political Economy of Open Source Software**. New York: Brooklyn College of the City University of New York, ano. Acesso em: <http://www.sci.brooklyn.cuny.edu/~schopra/chopra-dexter-TC05.pdf> Acesso em: 16 maio 2015

COLEMAN, Gabriella. **Coding Freedom**: the ethics and aesthetics of hacking. Princenton: Princeton University Press, 2013.

EVANGELISTA, Rafael A. **Traidores do movimento:** política, cultura, ideologia e trabalho no software livre. 2010. 250 f. Tese (Doutorado em Antropologia) – Instituto de Filosofia e Ciências Humanas, Universidade Estadual de Campinas, Campinas, 2010.

FISCH, Patrícia. **Redes Sociais e Educação:** elementos de design a partir da comunidade de software livre. 136 f. Dissertação (Mestrado em Tecnologia) - Universidade Tecnológica Federal do Paraná, Curitiba, 2008

GODBOUT, Jacques T. Introdução à dádiva. **Rev. bras. Ci. Soc.,** São Paulo, v. 13, n. pp. 39-52, out. 1998 . Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0102-69091998000300002&lng=pt&nrm=iso>. Acesso em: 21 jan. 2015.

GOETZ, Martin. **Memoirs of a Software Pioneer:** Part 1. IEEE Annals of the History of Computing, 2002. p. 43-56.

HOBSBAWM, Eric J. **A era dos extremos**: o breve século XX, 1914-1991. 2. ed. São Paulo: Companhia das Letras, 2009.

KON, Fábio; SABINO, Vanessa. **Licenças de Software Livre História e Características.** São Paulo: USP, 2009. (Relatório Técnico RT-MAC-IME-USP). Disponível em: <ccsl.ime.usp.br/files/relatorio-licencas.pdf> Acesso em: 21 mar. 2015.

KON, Fábio et al. **Software livre e propriedade intelectual:** aspectos jurídicos, licenças e modelos de negócio. Disponível em: <http://ccsl.ime.usp.br/files/slpi.pdf> Acesso em: 21 mar. 2015.

KON, Fábio; SABINO, Vanessa. **Licenças de Software Livre História e Características.** São Paulo: USP, 2009. (Relatório Técnico RT-MAC-IME-USP). Disponível em: <ccsl.ime.usp.br/files/relatorio-licencas.pdf> Acesso em: 21 mar. 2015.

LAKHANI, Karim R.; WOLF Robert G. **Why Hackers Do What They Do:** understanding motivation and effort in free/open source software projects. Boston: MIT Sloan School of Management, 2005. Disponível em: <http://ocw.mit.edu/courses/sloan-school-of-management/15-352-managing-innovation-emerging-trends-spring-2005/readings/lakhaniwolf.pdf> Acesso em: 21 mar. 2015

MAGALHÃES, Gildo. Das máquinas de calcular à informática. **Revista da SBHC,** n. 17, p. 21-28, 1997. Disponível em: <www.sbhc.org.br/arquivo/download?ID_ARQUIVO=203>. Acesso em 16 ago. 2014.

MARKOFF, John, **What the Dormouse Said.** how the sixties counterculture shaped the personal computer industry. London: Penguin Books, 2005.

MICROSOFT, **Microsoft takes .NET open source and cross-platform**, adds new development capabilities with Visual Studio 2015, .NET 2015 and Visual Studio Online. 2014. Disponível em: <https://news.microsoft.com/2014/11/12/microsoft-takes-net-open-source-and-cross-platform-adds-new-development-capabilities-with-visual-studio-2015-net-2015-and-visual-studio-online/>. Acesso: 20 de maio 2015.

RAYMOND, E. S. **The Cathedral and the Bazaar.** Würzburg: Linux Kongress, 1997. Disponível em: <http://www.catb.org/~esr/writings/cathedral-bazaar/> Acesso em: 10 de maio 2015.

RAYMOND, E. S. **Telling Lies:** ESR on Microsoft. 2008. Disponível em: <http://www.linuxjournal.com/article/5007>. Acesso em:10 de maio 2015.

RAYMOND, E. S. ***Goodbye, "free software"; hello, "open source".*** 1998. Disponível em: <http://www.catb.org/~esr/open-source.html>. Acesso em: 20 de mar. 2015.

SIMON, Imre; VIEIRA, Miguel .S. **A propriedade intelectual diante da emergência da produção social**. São Paulo: Universidade de São Paulo, 1997. (Texto de discussão)

SÖDERBERG, Johan. Copyleft vs. Copyright: A Marxist critique. **First Monday,** Volume 7 Number 3 - 4 March 2002. Disponível em:

<http://journals.uic.edu/ojs/index.php/fm/rt/printerFriendly/938/860> Acesso em 16 ago.

STALLMAN. Richard. M. **Free software, free society:** selected essays of Richard M. Stallman. 2nd ed. Boston: GNU, 2010. *softwares*

STALLMAN. Richard. M. **Why Open Source misses the point of Free Software.** Disponível em: <http://www.gnu.org/philosophy/open-source-misses-the-point.html.en>. Acesso em: 12 de maio de 2015.

TORRES, Aracele L. **A tecnoutopia do software livre:** uma história do projeto técnico e político do GNU. 2013. 206 f. Dissertação (Mestrado em História) – Faculdade de Filosofia, Letras e Ciências Humanas da Universidade de São Paulo, São Paulo, 2013.

TURNER, Fred. **From counterculture to cyberculture**: Stewart Brand, the Whole Earth network, and the rise of digital utopianism. Chicago: The University of Chicago Press, 2006.

VÄLIMÄKI, Mikko. **The Rise of Open Source Licensing:** a challenge to the use of intellectual property in the software industry. 2005. 264 p. Dissertation (Doctorate of Philosophy). Department of Computer Science and Engineering. Helsinki University of Technology. Helsinki. 2005.

YOST, Jeffrey R. **An interview with Martin Goetz.** Charles Babbage Institute, Center for the History of Information Processing, University of Minnesota, Minneapolis. 202. Disponível em: <http://conservancy.umn.edu/bitstream/11299/107328/1/oh334mg.pdf>. Acesso em 21 set. 2014.

WEBER, Steven. **The success of open source.** Boston: Harvard University Press. 2004