CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

PROGRAMA DE PÓS-GRADUAÇÃO EM EDUCAÇÃO TECNOLÓGICA

José Teixeira Horta Júnior

A FILOSOFIA DE SOFTWARE LIVRE DE RICHARD STALLMAN:
aspectos técnicos, éticos e pedagógicos

Belo Horizonte

2019

José Teixeira Horta Júnior

A FILOSOFIA DE SOFTWARE LIVRE DE RICHARD STALLMAN:
aspectos técnicos, éticos e pedagógicos

Dissertação apresentada ao Programa de Pós-Graduação em Educação Tecnológica, do Centro Federal de Educação Tecnológica de Minas Gerais como requisito parcial à obtenção do título de mestre em Educação Tecnológica.

Linha de pesquisa: Ciência, Tecnologia e Trabalho: Abordagens Filosóficas, Históricas e Sociológicas

Orientador: Prof. Dr. Luiz Henrique de Lacerda Abrahão

Belo Horizonte

2019

Horta Júnior, José Teixeira
H821f A filosofia de software livre de Richard Stallman: aspectos técnicos, éticos e pedagógicos. / José Teixeira Horta Júnior. – – Belo Horizonte, 2019.
109 f. : il.

Dissertação (mestrado) – Centro Federal de Educação Tecnológica de Minas Gerais, Programa de Pós-Graduação em Educação Tecnológica, 2019.
Orientador: Prof. Dr. Luiz Henrique de Lacerda Abrahão

Bibliografia

1. Ensino Profissional – Tecnológico. 2. Software Livre. 3. Stallman, Richard. I. Abrahão, Luiz Henrique de Lacerda. II. Centro Federal de Educação Tecnológica de Minas Gerais. III. Título

CDD 378.013

Elaboração da ficha catalográfica pela Bibliotecária Elisângela Gonçalves Barbieri CRB-6: 2624 / CEFET-MG

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
MESTRADO EM EDUCAÇÃO TECNOLÓGICA

José Teixeira Horta Júnior

"A FILOSOFIA DO SOFTWARE LIVRE DE RICHARD STALLMAN: aspectos técnicos, éticos e pedagógicos"

Dissertação apresentada ao Curso de Mestrado em Educação Tecnológica do Centro Federal de Educação Tecnológica de Minas Gerais - CEFET-MG, em 27 de junho de 2019, como requisito parcial para obtenção do título de Mestre em Educação Tecnológica, aprovada pela Comissão Examinadora de Defesa de Dissertação constituída pelos professores:

Prof. Dr. Luiz Henrique de Lacerda Abrahão - Orientador
Centro Federal de Educação Tecnológica de Minas Gerais

Prof. Dr. Marco Antônio Sousa Alves
Universidade Federal de Minas Gerais

Prof. Dr. José Geraldo Pedrosa
Centro Federal de Educação Tecnológica de Minas Gerais

À minha adorável esposa Érika,
pelo amor, carinho, paciência e inestimável colaboração.

AGRADECIMENTOS

Aos meus pais, José, pelo amor e cuidado, e Fátima, por todo o amor, carinho, cuidado, dedicação e inspiração.

À minha adorável esposa Érika, por todo o apoio e compreensão.

Às minhas filhas Jéssica e Júlia, por entenderem os momentos de ausência, pelos sorrisos, carinhos e incentivo.

À minha adorável avozinha Dona Antoninha, por me acordar todos os dias da minha infância para que eu fosse à escola.

Aos meus irmãos Jorsia, Kris, Karol, Pablo, Aline, Kaio e Jean, pela torcida, força e esperança.

Ao meu orientador Prof. Dr. Luiz Henrique de Lacerda Abrahão, pela direção, esforço, esclarecimento e boas ideias.

Aos membros da banca, Prof. Dr. José Geraldo Pedrosa e Prof. Dr. Marco Antônio Sousa Alves, pelas valiosas críticas e sugestões.

Aos professores do mestrado, pelos sábios ensinamentos transmitidos.

Ao meu grande amigo Prof. Carlos Marcos Alves, por todo apoio, incentivo e orientação.

“Não sou nada.
Nunca serei nada.
Não posso querer ser nada.
À parte isso, tem em mim todos os sonhos do mundo.”

Fernando Pessoa

RESUMO

Com o avanço tecnológico e a popularização do uso de computadores a partir do final dos anos 1990, diversos setores da sociedade foram influenciados e passaram a utilizá-los em suas atividades. Na escola, não é diferente, pois esse ambiente está cada vez mais envolvido com dispositivos tecnológicos e seus alunos e professores são altamente influenciados por eles. Por conseguinte, o aumento da utilização de tecnologias que, ao contrário de facilitar o desenvolvimento intelectual e a criatividade, submete o indivíduo a um processo de automatização, faz com que seja necessário repensar o uso de certas tecnologias em detrimento de outras. Assim, esta dissertação propôs estudo sobre quais tecnologias podem ser benéficas ao ambiente educacional, tendo como objetivo uma educação que seja emancipadora e que conduza o indivíduo à liberdade de pensamentos e ao desenvolvimento de um espírito comunitário. A fim de compreender as teses do fundador do GNU Project e da Free Software Foundation, Richard Stallman, referentes à filosofia do software livre e relacioná-la à educação profissional e tecnológica foram analisados documentos como, livros, artigos, palestras e vídeos com o método de análise de conteúdo. Buscou-se verificar também se o software livre representa uma filosofia que acredita que a liberdade de distribuição e de utilização de software é fundamental para o desenvolvimento e liberdade do ser humano. Ainda, procurou-se investigar se o uso de software proprietário em educação limita o potencial dos alunos e transmite a eles valores que prejudicam a sociedade. A utilização de tecnologia pode contribuir para a melhoria na educação e para a emancipação do indivíduo. No entanto, tecnologias que causem dependências devem ser evitadas neste ambiente. Isso impôs a constatação de que a utilização de software livre em ambientes educacionais pode contribuir para o bom desenvolvimento da educação e da liberdade individual, levando o ser humano à emancipação e propagando a educação cívica e o conhecimento coletivo.

Palavras-chave: Filosofia de software livre. Richard Stallman. Educação profissional e tecnológica. Software livre e educação profissional e tecnológica.

ABSTRACT

With the development of computing technologies and the popularization of computers since late 1990s, several sectors of society were influenced by it and started to make use of them in their activities. Education, for example, is increasingly making use of technological devices, as students and teachers are highly impacted by them. Consequently, the increase in the use of technologies in schools that, as opposed to nourishing intellectual development and creativity, submit the individual to an automation process which makes it necessary to rethink the use of certain technologies in detriment of others. Therefore, this dissertation proposed a study on how technologies can be beneficial to the educational environment, aiming at an education that is emancipating and leads to individual freedom of thought as well as to the development of a sense of community. In order to understand the theses of Richard Stallman, (founder of the GNU Project and the Free Software Foundation), books, articles, lectures, and content analysis method were analyzed focusing on the philosophy of free software and relating it to professional and technological education. It was also sought to verify whether the free software represents a philosophy which believes that freedom of distribution and the use of software are crucial for the development and freedom of a human being. Furthermore, we investigated whether the use of proprietary software in education could limit students to fulfill their potential and transmit values that may harm society. Technology can contribute to the improvement of education and to the promotion of individual emancipation. However, technologies that cause dependencies must be avoided in such environment. This research led to the realization that the use of free software in education can add to a good development in this area as well as grant individual freedom, creating ways in which a human being can become independent but also cultivates civic education and collective knowledge.

Keywords: Free Software Philosophy. Richard Stallman. Professional and technological education. Free software and professional and technological education.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O objetivo geral desta pesquisa é compreender as teses do fundador do GNU Project e da Free Software Foundation, Richard Stallman, referentes à filosofia do software livre e relacioná-la à educação profissional e tecnológica. Para tanto, foram delimitados os seguintes objetivos específicos: 1) identificar os personagens, as empresas e os acontecimentos que contribuíram para a revolução digital e permitiram o surgimento do software livre; 2) apontar e definir os diversos modelos de desenvolvimento e comercialização de softwares, suas características, vantagens e desvantagens; 3) analisar os fundamentos da filosofia do software livre de Richard Stallman; 4) relacionar a filosofia do software livre de Richard Stallman e a educação profissional e tecnológica.

O principal material utilizado na pesquisa foi a obra de Richard Stallman, composta de livros, artigos, entrevistas, documentação técnica e vídeos. Os livros de referência foram: *Software libre para una sociedad libre* (2004), *Free Software, Free Society* (2010) e *Free as in Freedom: Richard Stallman's Crusade for Free Software* (2002). Além desses, também foram utilizados textos sobre educação e sobre software livre de fontes primárias (como Paulo Freire, Linus Torvalds, Bill Gates) e comentadores (como Walter Isaacson e Sérgio Amadeu). Esses textos contribuíram para elucidar perguntas para quais não foi possível encontrar respostas nos estudos de Stallman. Todo trabalho de tradução dos textos foi realizado pelo autor da pesquisa.

Esta pesquisa está organizada em quatro capítulos principais, além desta introdução, das considerações finais e das referências. Essa organização foi escolhida para que fosse possível apresentar os principais acontecimentos, conceitos e termos antes que se pudesse aprofundar no tema principal da relação entre software livre e educação. No capítulo 2, "A revolução digital: seus personagens, empresas e acontecimentos históricos", foram apresentados os personagens, as empresas e os acontecimentos históricos que levaram ao surgimento da revolução digital, com a criação dos computadores pessoais, das licenças de software e dos softwares livres, além de apresentar qual a relação entre eles. Foram apresentados alguns personagens e suas criações que se destacaram durante esse período, como é o caso de Steve Jobs, Steve Wozniak, Bill Gates, Linus Torvalds e Richard Stallman. No capítulo 3, intitulado "Software livre e software

proprietário: definições iniciais", destacaram-se os modelos de desenvolvimento e comercialização de softwares e as questões técnicas e práticas a respeito de cada um dos modelos. Além disso, foram expostas algumas terminologias e definições necessárias para compreensão da discussão. No capítulo 4, denominado "Liberdade e controle sobre software", serão demonstrados os fundamentos da filosofia do software livre de Richard Stallman, suas implicações técnicas e filosóficas. Será discutida também a questão sobre o controle do desenvolvimento de software e a liberdade do seu uso. Por fim, no capítulo 5, intitulado "Contribuições do software livre para educação: educação cívica e emancipação do indivíduo", serão discutidas as contribuições que o software livre pode oferecer para a educação e para o ambiente educacional.

Essa organização permite que o leitor vá, aos poucos, familiarizando-se com a cultura *hacker*, para que possa depois entender o surgimento do movimento software livre e sua importância tanto para a evolução da computação em geral quanto para o mercado de tecnologia da informação. Por fim, o leitor poderá conhecer a relação entre software livre e educação.

Ao final, espera-se que o objetivo principal e os específicos tenham sido atendidos e a questão central elucidada, indicando, assim, que a utilização do software livre tende a contribuir para uma educação mais emancipadora, que valoriza o livre pensamento. Isso evidencia que a utilização de software livre pode conduzir o indivíduo a uma educação cívica em que ele tenha consciência de seu papel como parte de uma sociedade.

A relevância desta pesquisa residiria no fato de a investigação estar pautada na relação entre software livre e educação profissional e tecnológica e isso não ser uma abordagem usual. Portanto, acredita-se que os resultados desta pesquisa possam contribuir para as discussões no campo da educação profissional e tecnológica. Especificamente, sobre a relação entre uso de software nas escolas e o projeto de uma educação emancipadora.

Para fazer uma primeira aproximação ao tema do software livre, foi realizada uma pesquisa nos bancos de dados de teses e dissertações das instituições de ensino nacionais nos trabalhos produzidos entre 2002 e 2018, tais como a Biblioteca Digital Brasileira de Teses e Dissertações (BDTD) e o Catálogo de Teses e Dissertações da Capes, para analisar a literatura relacionada à educação, tecnologia e software livre. Com a busca das palavras-chaves "software livre na educação",

“Richard Stallman”, “filosofia *hacker*”, essa pesquisa resultou em 16 trabalhos. A seguir, apresenta-se uma breve revisão dessa literatura.

Zimmermann (2007) investigou, em sua pesquisa, a relação dos movimentos sociais com educação sob a perspectiva da comunicação intermediada por computador e sobre inclusão digital em um acampamento do Movimento dos Trabalhadores Rurais Sem Terra (MST). Ao final, o autor afirma que os movimentos sociais como MST se beneficiam do software livre como ferramenta e também como inspiração. Esses dois movimentos se assemelham, pois nos dois casos estão lutando contra a imposição das grandes indústrias, tanto em direitos autorais quanto em direito às propriedades improdutivas. O pesquisador coloca ainda que as desigualdades sociais são espelhadas nas oportunidades de acessos a sistemas de informação. Em suas conclusões, apresenta a tese de que a educação deve ser instrumento emancipador e que o software livre tem grande potencial para alavancar a autonomia técnica e viabilizar a educação como instrumento de cidadania.

Tavares (2007) realizou um estudo de caso para avaliar o processo de inovação no setor de software através do software livre como alternativa ao modelo tradicional de desenvolvimento. Essa pesquisa revelou que, apesar da questão de propriedade intelectual não ser o mais relevante no desenvolvimento de software livre, “[…] as expectativas de apropriabilidade com relação à inovação mantêm-se, apesar dos mecanismos de apropriabilidade tradicionais baseados em propriedade intelectual, segredo e patente, aparentarem ter sua importância minimizada neste novo modelo de negócio” (TAVARES, 2007, p. 89). Outra conclusão é que as empresas de software livre continuam a buscar um processo de inovação, apesar das mudanças ocorridas nos mecanismos de apropriabilidade por elas utilizados.

Moura (2007) investigou o uso de tecnologias da informação em um assentamento rural de trabalhadores sem terra e a possibilidade de o software livre contribuir para a produção e o conhecimento nessa comunidade. Ao final da pesquisa, o autor afirma que, como ferramenta de aquisição de conhecimento, o software livre contribui para a liberdade das comunidades pesquisadas. Contudo, a filosofia do software livre não é percebida pelos usuários. Esses usuários não tem a consciência da importância da utilização de desses softwares como instrumentos de libertação intelectual, assim como não percebem a semelhança entre a filosofia libertária do MST e a filosofia libertária do movimento software livre.

Já a pesquisa de Gomes (2007) investigou a questão do aprendizado e ensino utilizando-se software livre e suas relações com alunos com dificuldade de leitura e escrita. Tendo como objetivo verificar se a formação em informática por parte dos professores auxilia na melhoria de suas práticas docentes, sua pesquisa sugere que, em primeiro lugar, a formação do professor em informática precisa "trazer a aproximação das teorias com sua prática cotidiana e ocorrer no espaço escolar" (GOMES, 2007, p. 136). Essa formação, se apropriada, pode fazer com que o professor mude sua prática docente, melhorando assim a relação com esses alunos com dificuldade de leitura e escrita. O autor afirma ainda que, apesar da vontade dos professores de aprenderem, faltam incentivos por parte dos gestores públicos. Para ele, esses gestores devem abandonar os discursos sobre inclusão digital e criar ações efetivas para essa inclusão.

Em outra investigação, G. M. O. Souza (2008) trabalhou com as políticas de adoção do software livre pela Prefeitura Municipal de Fortaleza e suas influências na formação continuada dos professores dos laboratórios de informática daquela cidade. A autora pôde confirmar, ao fim de seu estudo, que, apesar da imposição por parte do poder público local e da iniciativa tardia na adoção de formação dos professores, a implantação de softwares livres nas escolas foi benéfica e influenciou a formação continuada dos professores. Afirmando ainda que essa formação necessita de aprimoramento, "para se alcançar um nível satisfatório de formação, e consequente prática profissional desses agentes do processo educacional" (SOUZA, 2008, p. 125).

Oliveira (2008) realizou estudo para analisar a formação de professores do ensino fundamental em escolas públicas municipais com centros de capacitação tecnológicas e com o uso de software livre na prática pedagógica. Destacou-se também a análise da formação do professor para uso de software livre e se essa formação está alinhada com as necessidades das práticas pedagógicas. A pesquisa tem como resultado a verificação de que essa formação oferecida aos professores não foi satisfatória, pois privilegiou os aspectos básicos de uso da tecnologia relacionado a certos aplicativos, de forma técnica e não pedagógica. O pesquisador conclui ainda que existe necessidade de criação de cultura de uso pedagógico desses softwares para que permitam atender às reais necessidades em sala de aula. "Com efeito, a utilização que se faz desses recursos é incipiente em relação à

gama de possibilidades que, infelizmente, não estão disponíveis na realidade dos saberes docentes dos professores pesquisados" (OLIVEIRA, 2008, p. 179).

A pesquisa de P. F. Souza (2008) analisou a aceitação do uso do software livre Muriqui na Casa do Professor de Ouro Preto-MG. Essa pesquisa avaliou a aceitação e a usabilidade desse software por parte dos alunos e professores daquela instituição. Como resultado, o pesquisador pode concluir que a utilização do software é viável para a prática pedagógica e para a capacitação dos professores para uso de computadores em sala de aula. No entanto, ele necessita de aprimoramento tanto nas funcionalidades quanto na comunicação com o usuário em fase de aprendizagem.

No texto de Darós (2011), foram trabalhados os conceitos de autonomia e de liberdade proporcionadas pelo uso de software livre nas práticas educacionais. O autor aborda também o fato de o software livre se desenvolver coletivamente, possibilitando a interação de diversas realidades, tornando assim o software mais diversificado e mais humanizado. Ao final, o pesquisador conclui que a escolha por tecnologias livres pode "proporcionar uma experiência de práticas libertadoras, com a possibilidade de que o indivíduo faça parte de todo o processo de maneira incondicional" (DARÓS, 2011, p. 72). Além disso, afirma que o entendimento das relações de dominação que as tecnologias podem ter sobre os indivíduos favorecerá a prática da educação. Isso permitirá a prática de inclusão digital de forma não excludente e possibilitará que a educação seja feita com compartilhamento e disseminação de resultados, tornando-os acessíveis à comunidade.

Ronzani (2011) faz em sua pesquisa uma avaliação do movimento software livre e as diversas formas em que esse movimento se dá, de acordo com cada país em que ele é desenvolvido. Além disso, ele analisa como as políticas locais e a importância que é dada à tecnologia no processo socioeconômico influenciam esse movimento localmente. O autor constata que um modelo híbrido de desenvolvimento de software, que na sua concepção seria um software com parte do código aberto, mas que ainda assim seria proprietário, é a tendência de mercado no futuro. Segundo ele, grandes empresas estão se aproximando do movimento software livre como estratégia de sobrevivência. No entanto, ele adverte que nem todos os países estão preparados para essa tendência, pois países menos desenvolvidos não estimulam o pioneirismo e a inovação, fazendo com que essas nações tenham uma relação de subordinação tecnológica aos países desenvolvidos.

Aguado (2012) fez uma análise das contribuições que o movimento software livre pode trazer à formação de redes de colaboração na educação, focando na possibilidade do processo dialógico ser construído a partir da autorreflexão que leva a tomada de consciência. Em suas considerações finais, o autor entende que o software livre tende a contribuir para a formação de redes de colaboração na educação. Segundo ele, a filosofia do software livre está intimamente relacionada à ideia de uma educação dialógica. Ele afirma ainda que as características das comunidades de software livre podem colaborar com a prática e com um processo dialógico e de busca de liberdade. Para isso, esse processo deve ser iniciado a partir de uma autorreflexão dos envolvidos nas redes de colaboração para que estes passem “a tomar consciência da realidade a qual desejam inferir” (AGUADO, 2012, p. 126).

Já Zílio (2013) estudou a relação entre as concepções de educação de professores da Rede Pública Estadual de Porto Alegre e a opção pelo software livre nas práticas pedagógicas. Esse estudo avaliou a real utilização desses softwares por parte dos professores. A autora conclui que o aparato tecnológico disponibilizado está sendo usado “[…] principalmente como mero recurso instrumental de práticas pedagógicas tradicionais, o que não representa um salto qualitativo na educação para a transformação social” (ZÍLIO, 2013, p. 120). Ela coloca como principal obstáculo para uma melhor utilização desse aparato o fato de que a maioria dos professores não está preparada para isso e que a implantação do sistema foi uma imposição e não um consenso. Reflete ainda que esses professores não estão conscientizados da importância da utilização de softwares livres em detrimento do software proprietário, e a razão para isso é esses professores não compreendem que “[…] a forma com que desenvolvemos nosso trabalho determina nosso modo de pensar” (ZÍLIO, 2013, p. 120)

A pesquisa realizada por Cortes (2013) teve como objetivo apresentar a trajetória do desenvolvimento dos computadores e dos videogames levando-se em conta a filosofia *hacker* e a posterior apropriação desses equipamentos pelos meios corporativos. Destaca-se ainda a formação de grupos sociais e ativistas virtuais a partir dessas apropriações. Por fim, o autor apresenta o argumento de que a população brasileira é despolitizada e está acostumada a receber as notícias dos meios de comunicação sem nenhum questionamento. O autor vê, na cultura dos usuários de internet e na cultura *hacker*, uma forma de fugir dessa alienação.

Segundo ele, as novas gerações estão costumadas com essa cultura colaborativa que “[…] se encontra enraizada no universo dos computadores, em especial na internet, mas também no mundo dos jogos” (CORTES, 2013, p. 112). Ele afirma também que o projeto Jogo Justo não é um movimento ciberativista nem mesmo uma ação política tradicional, pois o tipo de ativismo praticado no projeto seria um “ativismo de sofá” e uma espécie de ilusão ciber-utópica.

Já Andrade (2013) trabalha a questão da inclusão digital e a potencialização do aprendizado através do uso de software livre, dentro do âmbito do programa “Um computador por aluno”, do governo federal. Em suas considerações finais, afirma que, em razão das relações de mediação dos sujeitos do conhecimento com o mundo, o acesso a tecnologias é fundamental para o desenvolvimento das diversidades culturais com influências na cultura escolar. Ele diz ainda que as ferramentas técnicas interferem no desenvolvimento cognitivo das pessoas e na construção da cultura e por isso devem ser apropriadas livremente. Segundo ele, a cultura digital está em processo de formação. Ele afirma que os professores envolvidos no projeto “Um computador por aluno” não possuíam habilidades suficientes para operação dos equipamentos e que as dificuldades provenientes dessa inabilidade foram a causa da resistência ao projeto. Ao final, conclui que, mesmo com as dificuldades apresentadas, é possível que grupos de compartilhamentos surjam formando comunidades de usuários de softwares educacionais livres, com a possibilidade de criação de um contexto criativo “[…] para uma apropriação tecnológica do jeito e com os conteúdos que a escola precisa – com o uso consciente das tecnologias digitais na educação e com autonomia tecnológica capaz de interferir de forma libertária na realidade individual e coletiva” (ANDRADE, 2013, p. 159)

A pesquisa de Galvão (2014) teve como foco a apresentação de um software livre para simulação de ambiente robótico educacional e trata de questões relacionadas à utilização desse software. O pesquisador conclui que o uso do software simulador “[…] mostrou-se de grande importância no processo de abstração do conhecimento relacionado à lógica de programação, por proporcionar a seus usuários uma experiência visual bidimensional que antecipa a ação que será seguida pelo robô” (GALVÃO, 2014, p. 87).

Carvalho (2015) analisa o uso de software livre nas escolas públicas e no ensino superior através da experiência da utilização do programa Scribus no curso

de Jornalismo da Universidade Federal de Uberlândia (UFU). O autor destaca a imposição do uso de software proprietário através de estratégias comerciais em detrimento do uso de software livre. O trabalho dá ênfase ao caráter libertador e ao desenvolvimento do espírito solidário e colaboracionista que o software livre proporciona. Nas considerações finais, o autor relembra que "[…] a filosofia de software livre apresenta ideias liberais, de participação e compartilhamento do conhecimento, que poderiam ser reforçadas na universidade pública […]", porém a cultura do software proprietário acaba se impondo. Foi possível observar um aumento no interesse pelas atividades desenvolvidas pela disciplina, mas alguns alunos entraram em conflito por causa do direcionamento do mercado que lhes impõe o uso de software proprietário (CARVALHO, 2015, p. 62).

O estudo de Souza (2016) analisou quais as contribuições de um curso de formação continuada na visão dos professores do Ensino Fundamental I sobre a utilização do software Gcompris como ferramenta de ensino em sala de aula, de uma escola da educação básica no município de Boa Vista-RR. A conclusão apresentada pelo autor da pesquisa foi a de que "a realização das atividades feitas com os professores proporcionou uma visão da necessidade da formação continuada para acompanhar a imposição social sobre o uso das tecnologias" (SOUZA, 2016, p. 111). Foi possível perceber que o software estudado proporciona uma aprendizagem que pode ser prazerosa e significativa. Essa impressão é corroborada pela opinião de professores na formação continuada e na utilização do software por eles com seus alunos.

Diante da produção acadêmica relativa à educação profissional e tecnológica e software livre encontrada nas pesquisas realizadas, verifica-se que tais textos não investigaram a relação entre esses dois temas. As análises foram realizadas de forma isolada e em casos específicos, com estudos de caso. Nenhum dos textos encontrados relaciona software livre às questões técnicas, éticas, ideológicas e pedagógicas, e essas não são diretamente relacionadas entre si. Além disso, foi possível constatar que as discussões estão segmentadas em questões práticas e em questões filosóficas. No entanto, elas são tratadas, na maioria das vezes, separadamente e de forma superficial.

O autor no qual esta pesquisa se baseia, Richard Stallman, aparece nos demais textos acadêmicos estudados, mas sem muito aprofundamento das ideias propostas por ele. Não foi possível identificar nesses documentos uma análise

detalhada da obra do autor. Sua produção, que trata dos assuntos sociais relacionados à computação, à educação e à liberdade, é o foco de exame desta pesquisa. O que se propõe é a exploração das teorias do autor nas áreas de ciências da computação, filosofia de software livre, ética e educação e verificar se existe relação entre essas áreas.

Em consequência da dificuldade em encontrar pesquisas que investiguem a relação do software livre com a educação, formulou-se a questão que norteou este estudo: “Que relações podem ser pensadas entre a filosofia do software livre de Richard Stallman e a educação profissional e tecnológica?”.

Para responder à questão formulada, foi feita uma pesquisa teórica sobre a filosofia de software livre e sua relação com a educação profissional e tecnológica, que é “[…] dedicada a reconstruir teoria, conceitos, ideias, ideologias, polêmicas, tendo em vista, em termos imediatos, aprimorar fundamentos teóricos” (DEMO, 2000, p. 20). Para Marcela Cavalini (2016), “pesquisa teórica, pressupõe a discussão e fundamentação da teoria além de dar margem à possíveis contra-argumentos e questionamentos acerca da legitimação das hipóteses”.

A técnica utilizada para o exame do material foi a análise de conteúdo “compreendida como um conjunto de técnicas de pesquisa cujo objetivo é a busca do sentido ou dos sentidos de um documento” (CAMPOS, 2004). Com isso objetiva-se compreender, de forma global, a obra de Richard Stallman sobre software livre e educação. Bardin (1977 *apud* CAMPOS, 2004) afirma que

> […] a análise de conteúdo se constitui de várias técnicas onde se busca descrever o conteúdo emitido no processo de comunicação, seja ele por meio de falas ou de textos. Desta forma, a técnica é composta por procedimentos sistemáticos que proporcionam o levantamento de indicadores (quantitativos ou não) permitindo a realização de inferência de conhecimentos.

Com efeito, esta pesquisa não teve como objetivo mapear questões práticas relacionadas a projetos pedagógicos específicos ou implantação de sistemas em ambiente educacional. Portanto, cabe ainda investigar a utilização dos conceitos aqui estudados de forma prática e em sala de aula.

2 A REVOLUÇÃO DIGITAL: SEUS PERSONAGENS, EMPRESAS E ACONTECIMENTOS HISTÓRICOS

Este capítulo visa contextualizar o surgimento da cultura *hacker* e apresentar o papel de algumas pessoas e empresas que foram diretamente responsáveis pelo grande avanço ocorrido na computação nos últimos 60 anos. Para isso, serão expostos, nas seções seguintes, eventos históricos com o objetivo de demonstrar os diferentes interesses e abordagens de algumas figuras mostradas. Além disso, o capítulo irá indicar a conjuntura que levou ao fim da comunidade *hacker*, originalmente formada no Massachusetts Institute of Technology (MIT), e ao aparecimento de novas comunidades, como as de software livre e de *open source*.

2.1 Revolução digital

A partir dos anos de 1960, acontecimentos históricos no mundo da computação desencadearam o início do que veio a ser conhecido como a revolução digital. Segundo Isaacson (2014, p. 15), “historiadores da ciência às vezes são avessos a chamar de revoluções períodos de grandes mudanças”. No entanto, outros historiadores estão convencidos de que a revolução digital é tão real quanto outras revoluções do passado. Moraes e Kohn (2007, p. 4) afirmam que:

> Nos primórdios da tecnologia humana, passou-se da atividade agrária para a industrialização das cidades, por conseguinte, esse processo mudou a estrutura social de forma tão ampla que foi denominada revolução. Do mesmo modo, as transformações ocorridas com o desenvolvimento tecnológico podem ser consideradas uma revolução contemporânea da ascensão digital e da informação.

Nesse sentido, o grande avanço experimentado pelas ciências da computação se apresentou para os que se dedicam às tecnologias de informação como uma revolução. Isso se deu porque alterou totalmente os meios de comunicação, levando o homem a uma interação mundial jamais experimentada. Sendo possíveis, por exemplo, conversas via videoconferência em tempo real por pessoas a milhares de quilômetros de distância umas das outras. Reuniões podem ser feitas remotamente, funcionários e equipamentos podem ser monitorados e operar através da internet, transações bancárias podem ser realizadas por meio de telefones celulares ou sites. Conforme afirma Silveira (2004, p. 42), “as tecnologias

de informação e comunicação estão se consolidando como meios de expressão do conhecimento, de expressão cultural e de transações econômicas".

Essa revolução alterou também o modo como a maioria das empresas fazem negócios. Foram criados novos mercados virtuais e físicos, os quais estão voltados para mercadorias reais, tais como livros, celulares, computadores, alimentos e para mercadorias virtuais e serviços diversos, a exemplo de sites de relacionamentos, sites de empregos, delegacias e cartórios virtuais, treinamentos e faculdades a distância etc.

Durante grande parte do início dessa revolução, principalmente entre os anos 1950 e 1970, estavam sendo desenvolvidas pesquisas em diversas universidades do mundo. Entretanto, a maior parte de tais pesquisas estava voltada para o desenvolvimento de tecnologia para uso militar ou para grandes empresas, como IBM, AT&T e Bell. Ainda assim, havia um grupo de aficionados por eletrônica que nutriam seus próprios motivos para participarem dessa revolução. Eram pessoas de todo o mundo que amavam estudar eletrônica – e por isso eram chamados de amadores ou "hobbistas"[1]. Essas pessoas estavam mais interessadas em se divertirem do que ganharem dinheiro com seus projetos. Por isso mesmo, construíam em suas garagens protótipos com intuito de mostrá-los aos outros e compartilharem suas descobertas, por pura diversão. Conforme diz o historiador Isaacson: "Com a paixão de hobbistas, queriam ter seu próprio computador – não um aparelho compartilhado, ou em rede com outras pessoas, mas um aparelho com o qual pudessem brincar sozinhas, no quarto ou no porão" (ISAACSON, 2014, p. 318). Os amadores se reuniam em feiras e convenções onde eles trocavam informações sobre assuntos de interesse comum. Apresentavam suas ideias e seus projetos para outros aficionados sem nenhum receio de serem copiados. Havia nisso tudo uma solidariedade no sentido de compartilhamento de conhecimento. Desse grupo de amadores surgiram figuras que hoje são conhecidas por suas contribuições para a computação, embora alguns não se conhecessem inicialmente e tivessem suas contribuições em períodos diferentes. A esse exemplo, podem ser citados Richard Stallman, Bill Gates, Steve Wozniak, Steve Jobs, Linus Torvalds, entre outros.

[1] O termo "hobbista", largamente em uso, vem do inglês *hobbyist*, nome que se dá a uma pessoa dedicada a um *hobby*. Site Corte Certo (2015).

As grandes empresas de computação, como a IBM, Intel, AT&T e Bell, além de grandes universidades como MIT e Universidade da Pensilvânia, nos Estados Unidos, e das universidades de Manchester e de Cambridge, na Inglaterra, através de seus departamentos de pesquisa, detinham o monopólio do conhecimento e do desenvolvimento em computação. Esse monopólio continuou até a época do lançamento de computadores de pequeno porte e, mais tarde, dos computadores pessoais, como o Altair[2] 8800. Isso se devia ao fato de que

> […] os dispositivos computacionais ainda estavam confinados ao público especializado – os "cérebros eletrônicos", como eram chamados pela imprensa, ainda eram desconhecidos da população em geral, muito devido aos altos custos destes equipamentos, e a necessidade de conhecimento técnico para a sua operação (PERANI, 2016, p. 2).

A programação desses computadores era feita através de terminais e o acesso era controlado por tempo de computação. Para a gravação de programas, eram utilizados cartões perfurados e fitas magnéticas. A ideia de se produzir computadores pessoais era vista com desconfiança por empresas como a IBM e a Digital Equipment Corporation (DEC). Ken Olson, presidente e fundador da DEC, disse o seguinte a esse respeito: "Não há nenhuma razão para alguém querer um computador em casa" (PC WORD, 2008).

Com o lançamento do Altair, os amadores puderam ter acesso ao hardware de um computador. Para obter conhecimento a respeito dessas máquinas, eles buscavam informações em revistas especializadas em eletrônica, que disponibilizavam aos leitores informações sobre como conseguir parte do conhecimento sobre construção de computadores rudimentares para seus experimentos. Uma dessas revistas era a *Popular Electronics*. Conforme afirma o historiador Isaacson (2014), esses apaixonados por eletrônica,

> […] aliados a *hippies* do *Whole Earth Catalog*[3] e *hackers* caseiros, lançaram uma nova indústria, a dos computadores pessoais, que impulsionaria o crescimento econômico e transformaria nosso modo de viver e trabalhar. Num movimento de transferência de poder para o povo, os computadores foram tirados do controle exclusivo das corporações e dos militares e postos nas mãos de indivíduos, sendo convertidos em ferramentas de

[2] O Altair, desenvolvido pela empresa Micro Instrumentation and Telemetry Systems (MITS), era um computador de pequeno porte, que não tinha nenhum dispositivo de entrada ou de saída de dados. O resultado de seu processamento era mostrado através de luzes. Esse computador estava voltado para os amadores de eletrônica e deveria ser montado por eles (PERANI, 2016).

[3] *Whole Earth Catalog* foi um catálogo americano de contracultura desenvolvido entre 1968 e 1972, e ocasionalmente em outros anos antes de 1998 (WIKIPÉDIA, 2017).

> enriquecimento, produtividade e criatividade pessoais (ISAACSON, 2014, p. 318).

Entre os amadores, havia grupos diferentes e que se interessavam em aspectos também diferentes da computação. Enquanto alguns estavam interessados no hardware dos computadores e em como construí-los, como era o caso de Steve Wozniak e Steve Jobs, outros se interessam em como programá-los, no software, a exemplo de Bill Gates e Richard Stallman.

Jobs e Wozniak contribuíram para a popularização do computador pessoal e fundaram a Apple Computer em 1976, empresa de computação que detém o público consumidor mais apaixonado. A cada lançamento da empresa, consumidores do mundo inteiro fazem filas para comprar os seus produtos. A Apple revolucionou não só a indústria de computadores pessoais, com o lançamento do Apple II, em 1977, mas também a de telefones móveis, com o lançamento do seu *smartphone*, o iPhone, em 2007, e a de músicas, com o lançamento do iPod, que é um aparelho reprodutor de músicas em formato digital, em 2001 (KAHNEY, 2008).

Steve Wozniak deixou a Apple em 1985, depois de um acidente de avião e um período de amnésia. Em entrevista para a revista *Época Negócios*, ele disse o seguinte sobre sua saída:

> Quando me recuperei, percebi que era a última chance de ter um diploma de faculdade. A empresa tinha crescido, havia muitos engenheiros e um sistema de gerenciamento completo. A Apple seria bem-sucedida sem mim, e eu podia usar minhas ações para financiar outros projetos. Cheguei a voltar para a empresa para ajudar na divisão do Apple II e III. Mas saí de novo. Amo estar em grupos pequenos que falam sobre ideias e coisas que nunca existiram. Fui então fazer o primeiro controle remoto universal, que é usado até hoje. Depois, dei aula e fui para a filantropia. Tinha US$ 120 milhões em ações da Apple quando saí. Doei boa parte. Muito também se foi em meus dois divórcios. Como as ações hoje valem bem mais, minhas ex-exposas devem estar bilionárias (BARIFOUSE, 2018).

Steve Jobs esteve afastado da Apple por mais de dez anos, por causa de problemas de relacionamento com os funcionários e acionistas. Além da Apple, Jobs ainda foi responsável pelo sucesso da Pixar, empresa de animação que comprou em 1986 da Lucasfilms. A Pixar levou a indústria de desenhos animados a dar um salto enorme de qualidade com o filme *Toy Story*, de 1995, o primeiro totalmente produzido por computação gráfica. Jobs retornou à Apple em 1996, mesmo ano em que vendeu para a empresa o seu sistema operacional chamado NEXT. Esse retorno se deu pelos problemas financeiros vividos pela Apple. Em 1997, Jobs

assumiu definitivamente a chefia da Apple e iniciou uma reformulação e uma reaproximação com a Microsoft, conseguindo dessa empresa um apoio de seu presidente, Bill Gates, e um investimento de 150 milhões de dólares. Outra estratégia foi a de valorização da marca Apple com campanhas de publicidade. Com essas e outras ações, a Apple se recuperou e se consolidou como marca e empresa (KAHNEY, 2008). Jobs faleceu em 2011 por complicações de um câncer de pâncreas.

No período inicial em que a computação estava sendo descoberta pelos amadores, os que estavam interessados em software se dividiram, e um movimento com ideias de liberdade ainda mais profundas, os *hackers*, surgiu. Esses programadores se interessavam (e ainda se interessam) não apenas pelo aspecto técnico, mas também pela democratização do conhecimento e pela liberdade de pensamento. Essas ideias eram oriundas da subcultura *hacker*, que "surgiu do *Tech Model Railroad Club* do MIT, uma associação de estudantes *geeky*[4] – aficionados de tecnologia, jogos eletrônicos, quadrinhos etc." (ISAACSON, 2014, p. 214).

Esse grupo, que mais tarde se interessaria também por computação, convivia em um ambiente de compartilhamento de ideias e informações. Segundo o MIT, o termo *hacker* estava ligado à criatividade e a conhecimentos técnicos e espírito lúdico, brincadeiras e diversão. Conforme explicação no site da própria instituição: "No MIT, um *hacker* é alguém que faz algo intensamente, de forma interessante e criativa. Isso se aplica a qualquer coisa, desde escrever programas de computador até realizar uma brincadeira inteligente que diverte e encanta a todos no campus" (MIT, 2007). Essa definição é totalmente diferente da atual, que trata o termo de forma pejorativa e que liga os *hackers* a roubo de informações, sequestro de dados e invasões ilegais. O termo *hacker* sofreu um ataque difamatório,

> […] por volta de 1980, quando a mídia de notícias tomou conhecimento dos *hackers*, eles se fixaram em um aspecto estreito do *hacking* real: a segurança que o *hacker* ocasionalmente fazia. Eles ignoraram todo o resto do *hacking* e levaram o termo a significar quebrando a segurança, nada mais e nada menos. A mídia já espalhou essa definição, desconsiderando nossas tentativas de corrigi-las. Como resultado, a maioria das pessoas tem

[4] A palavra *geek* diz respeito a pessoas com especial interesse em tendências tecnológicas, computadores e dispositivos eletrônicos em geral. De modo genérico, pode-se observar a prevalência de profissionais (ou aspirantes a profissionais, no caso dos mais jovens) ligados às áreas de comunicação, ciência e tecnologia. Normalmente são fãs de tecnologia, eletrônica, jogos eletrônicos ou de tabuleiro, histórias em quadrinhos, livros, filmes, animes e séries (YOKOTE, 2014, p. 60).

> uma ideia equivocada do que os *hackers* realmente fazem e o que pensam (STALLMAN, 2017).

No departamento de computação do MIT, assim como outros de outras universidades, havia um grupo de programadores: *hackers* que desenvolviam softwares para resolver os problemas práticos do dia a dia e faziam pesquisas na área de computação. Nesse departamento havia uma impressora com problemas de travamento, o que levou os programadores a criaram softwares para contorná-los. No entanto, o departamento ganhou da empresa Xerox uma nova impressora que também passou a apresentar problemas. Porém, dessa vez, eles tinham assinado termos que os proibiam de desenvolver softwares para a impressora. Além disso, a Xerox não disponibilizava o código-fonte do programa que controlava a impressora (ISAACSON, 2014).

Richard Stallman era um dos programadores do MIT nessa época, e os problemas com a licença da Xerox fizeram com que ele refletisse sobre o desenvolvimento de softwares e o direito de propriedade desses softwares. Esse evento, aliado a outros que se seguiram, transformaram Stallman. Ele passou a questionar todo o sistema de produção de software que se iniciava e começou a protestar contra o controle das grandes corporações. Segundo o próprio Stallman, sua defesa intransigente de seus ideais fez com que muitas pessoas o comparassem com um profeta: “Algumas pessoas me comparam a um profeta do Antigo Testamento, e o motivo é que os profetas do Antigo Testamento diziam que certas práticas sociais estavam erradas. Eles não faziam concessões em questões morais” (WILLIAMS, 2002, p. 121). Os ideais de Stallman e suas contribuições para a computação serão objetos de estudos dos capítulos 3, 4 e 5: “Software livre e software proprietário: definições iniciais”, “Liberdade e controle sobre software” e “Contribuições do software livre para educação: educação cívica e emancipação do indivíduo”, respectivamente.

As próximas seções apresentam um breve histórico de três dos personagens mais importantes da revolução digital que estão ligados ao software, são eles: Bill Gates, Richard Stallman e Linus Torvalds.

2.2 Bill Gates

Enquanto a indústria de computação estava em pleno desenvolvimento, um jovem estudante, que ainda estava na escola secundária, também se desenvolvia e tornava-se um prodígio da computação destacando-se e iniciando um movimento diferente do que havia no MIT.

William Henry Gates III, conhecido como Bill Gates, nasceu em 1953, em Seattle, nos Estados Unidos, e era filho de um casal de classe média alta. O pai, William H. Gates, era advogado e a mãe, Mary Maxwell Gates, era líder civil; dessa maneira, pôde ter todas as oportunidades que alguém de sua posição social poderia ter.

Gates se destacou muito cedo na escola, tanto para matemática quanto para computação. De personalidade forte, era considerado rebelde pelos pais e professores. Por causa disso, seus pais o levaram a um psicólogo para tentar entender o motivo de sua rebeldia, porém, depois de um ano de terapia, foram convencidos de que não poderiam mudar seu temperamento. Ele era um garoto que gostava de esportes individuais e xadrez. Além de ter sido escoteiro e recebido várias insígnias. Por causa de seu destaque na escola, quando ele estava para ingressar na sétima série, seus pais decidiram colocá-lo em uma escola particular e para isso escolheram a escola Lakeside. Foi essa escolha que o colocou no caminho dos computadores (SMITH, 2016).

Nessa escola, a vida de Gates sofreu uma mudança com a chegada de um terminal de computador. “O clube de mães de Lakeside decidiu destinar o dinheiro arrecadado em um bazar de caridade à instalação de um terminal e à contratação de tempo de computador para os alunos” (GATES; MYHRVOLD; RINEARSON, 1995, p. 24). Esse terminal foi acondicionado em uma pequena sala do Edifício de Ciências e Matemática e foi apelidada de “Sala do Computador”. Para Gates e demais alunos, essa atitude do clube de mães foi extraordinário. Conforme suas próprias palavras: “Permitir que colegiais usassem um computador, no final dos anos 60, foi uma ideia extraordinária para Seattle da época – pela qual serei eternamente grato” (GATES; MYHRVOLD; RINEARSON, 1995, p. 24). Depois que a verba do clube de mães se esgotou, Gates e Paul Allen, um amigo que conheceu na Sala do Computador, buscaram outras formas de acesso a tempo de computador. Eles, juntamente com outros estudantes, fundaram um clube de computação. Segundo Gates afirmou em

seu livro, *A estrada do futuro*, seus pais pagavam a escola e os livros, porém eles precisavam se virar para conseguir tempo de computador. Por isso, ele e Allen começaram a trabalhar como programadores de software assistentes, recebendo cerca de 5 mil dólares, parte em dinheiro e o restante em tempo de computador, durante o verão. Esse trabalho deu a Gates e a Allen a compreensão necessária sobre sistemas operacionais e funcionamento de computadores. Em sequência, eles conseguiram também outros contratos para descobrir problemas em computadores em troca de poder utilizá-los. Basicamente, o que tinham de fazer era tentar travá-los (GATES; MYHRVOLD; RINEARSON, 1995).

Bill Gates se candidatou para os exames de admissão de três universidades, Harvard, Yale e Princeton, e foi admitido nas três. No entanto, acabou escolhendo matemática em Harvard. Na universidade, o fundador da Microsoft se dedicou à matemática aplicada e conseguiu se destacar razoavelmente nessa área. Contudo, ele continuou interessado em computadores e passava horas no laboratório de computação. Este laboratório era financiado pelas Forças Armadas, que haviam enviado um computador PDP-10, que inicialmente estava destinado para a Guerra do Vietnã, mas havia sido realocado para ajudar as pesquisas em Harvard, fato que não era comentado. Até por isso, não existiam regras escritas para o uso do computador. Em seu primeiro ano em Harvard, Gates utilizava o laboratório para projetos pessoais voltados para diversão. Aproveitando-se de que havia ainda vários computadores PDP-1, em que era possível, para os alunos, jogar videogames, Gates conectou esses computadores ao PDP-10 para criar um jogo de beisebol. Para terminar o jogo, ele passava horas no laboratório e ficava até tarde da noite escrevendo os algoritmos para controlar a bola e os saltos (ISAACSON, 2014).

Quando Bill Gates estava no segundo ano em Harvard, Paul Allen tentou convencê-lo a deixar o curso de matemática e aceitar um emprego na empresa na qual ele trabalhava, a Honeywell, mas, sob pressão dos pais, Gates não o fez. Allen já avia deixado seu curso na Universidade Estadual de Washington, sob influência de Gates. Em dezembro de 1974, Allen e Gates estavam verificando revistas em uma banca de jornal quando encontraram a edição de janeiro da revista *Popular Electronics*, que estampava o computador Altair na capa. Gates descreveu o momento da seguinte forma:

> Quando vimos aquilo, entramos em pânico. “Não, está acontecendo sem a gente. Vão começar a escrever software de verdade para esse *chip*.” Eu tinha certeza de que aconteceria, e rápido, e queria estar envolvido desde o início. Participar da revolução da microinformática desde os estágios iniciais parecia minha grande oportunidade, e não deixei escapar (GATES; MYHRVOLD; RINEARSON, 1995, p. 60).

Assim, juntamente com Allen, convenceram o proprietário da empresa Micro Instrumentation and Telemetry Systems (MITS), situada em Albuquerque, Estados Unidos, Ed Roberts, a licenciar um sistema criado por eles para inclusão em todos os computadores Altair. O problema é que eles ainda não tinham um sistema. Para isso eles escreveram o código de um interpretador de comandos para o BASIC[5]. Esse interpretador seria, mais tarde, conhecido como Microsoft Basic e seria considerada a primeira linguagem de programação comercial nativa de alto nível para microprocessador. A linguagem de programação Microsoft Basic inaugurou a indústria do software para computadores pessoais e o conceito de licença de uso (ISAACSON, 2014).

2.2.1 Gates x hobbistas

Havia um problema com o código do BASIC de Gates e Allen. Ele teria sido desenvolvido em computadores do departamento de defesa e, além disso, Gates havia permitido que uma pessoa que não era aluno utilizasse os computadores com sua senha. Por isso, Gates passou por um processo disciplinar movido pela universidade, porém foi absolvido. No entanto, eles foram obrigados a liberar uma versão do BASIC para domínio público. Esse fato o colocaria em uma posição vulnerável durante uma disputa com os hobbistas no clube de computação, que o acusariam de utilizar recursos públicos para criar um sistema que ele estava vendendo. Nesse episódio, cópias do BASIC foram disponibilizadas de forma gratuita, através de fitas, em uma reunião do Homebrew Club. Ao tomar conhecimento, Gates escreveu, enfurecido, cartas direcionadas aos hobbistas questionando o fato de eles estarem usando seu software sem que ele fosse recompensado pelo esforço de criação do programa. Conforme ele mesmo contou depois:

[5] BASIC é uma linguagem de programação. Ela foi criada pelos professores John George Kemeny, Thomas Eugene Kurtz e Mary Kenneth Keller, em 1964, no Dartmouth College, para fins didáticos.

> Escrevi uma carta aberta amplamente divulgada pedindo aos primeiros usuários de computadores pessoais que parassem de roubar nosso software para que pudéssemos ganhar dinheiro que nos permitiria elaborar novos programas. “Nada me daria mais prazer do que poder contratar dez programadores e inundar o mercado de *hobby* com software de boa qualidade”, escrevi. Mas meu argumento não convenceu muita gente a pagar por nosso trabalho; todos pareciam gostar dele e usá-lo, mas preferiam “tomar emprestado” a comprá-lo (GATES; MYHRVOLD; RINEARSON, 1995, p. 120).

Em resposta, os hobbistas escreveram algumas centenas de cartas ofensivas e as enviaram a Gates. Além disso, publicaram um boletim que sugeria que Gates havia desenvolvido o BASIC utilizando tempo de computador financiado com dinheiro público (ISAACSON, 2014). Esses acontecimentos distanciaram Gates do movimento *hacker* e o colocou ao lado das grandes corporações e o tornou inimigo desse movimento.

Em 1975, faltando apenas dois semestres para conclusão de seu curso em Harvard, Gates decidiu se mudar para Albuquerque e abandonou a faculdade. “Na primavera de 1975, Paul deixou o emprego de programador e decidi pedir licença de Harvard” (GATES; MYHRVOLD; RINEARSON, 1995, p. 64). Em Albuquerque, eles mantiveram o fornecimento de software para o Altair com um novo tipo de licença que não dava a propriedade ao MITS, mas sim o direito de usar o BASIC em seus computadores, pagando *royalties* por computadores vendidos com o software. Além disso, o MITS deveria tentar vender o BASIC para outros fabricantes de computadores. Com essa estratégia de negócios inovadora, Gates, Allen e outros dois estudantes fundaram a Microsoft e mudaram a forma de fazer negócios com computadores (GATES; MYHRVOLD; RINEARSON, 1995).

A Microsoft se tornou a maior empresa de computação da história. E mais de 40 anos depois ainda é líder no mercado de software e uma das maiores no mercado de computação em nuvem. Segundo a revista *Computerworld* (2017), em 2017 o Microsoft Windows representava 91,5% dos sistemas operacionais instalados. A Microsoft é também pioneira em pesquisa de computação voltada para usuários de computadores domésticos.

A licença de software é a chave para o sucesso financeiro da Microsoft. Essa licença permite ao usuário a utilização do software para os fins contratados, porém não transfere a ele a sua propriedade. Nesse sentido, o usuário não pode modificá-lo para outros fins. Essa licença também não permite a transferência do direito de

uso para outro usuário. Por trás da licença de software, existe a ideia de que o software é propriedade de quem o desenvolveu.

Este conceito de software proprietário será estudado no capítulo 3, “Software livre e software proprietário: definições iniciais”.

2.3 Linus Torvalds

No início nos anos 1990 e no auge da revolução digital, um jovem estudante criou um sistema que marcou uma nova etapa na história da computação. Linus Benedict Torvalds é um *hacker* e cientista da computação finlandês naturalizado americano, conhecido mundialmente pela criação do Linux. Torvalds nasceu em 1969, em Helsinque, na Finlândia. Seus pais são um jornalista e radialista ligado ao partido comunista chamado Nils Torvalds e uma editora gráfica chamada Anna Torvalds. Linus Torvalds foi criado ao lado da mãe e de sua irmã, Sara Torvalds, que também trabalha com jornalismo. Ele tem também um meio-irmão, chamado Leo Torvalds, fruto do relacionamento de seu pai com outra mulher. Seus pais se separaram, e Linus, Sara e sua mãe passaram por dificuldades financeiras. A família vivia se dividindo, com Sara se mudando para casa do pai e voltando para casa da mãe algumas vezes. Sua mãe tinha que trabalhar e por isso, muitas vezes, ele e a irmã ficavam sozinhos em casa e tinham que preparar a própria comida (TORVALDS; DIAMOND, 2001).

Torvalds era um garoto pouco confortável com sua aparência, como ele mesmo disse:

> […] eu era um *nerd*. Um *geek*. Desde muito cedo. Eu não colei meus óculos com fita adesiva, mas eu poderia muito bem ter feito isso, porque eu tinha todos os outros traços. Bom em matemática, bom em física, e sem graça social nenhuma. E isso foi antes que ser um *nerd* fosse considerado uma coisa boa (TORVALDS; DIAMOND, 2001, p. 3).

Na escola, quando jovem, sempre obtinha bons resultados, não porque se esforçasse, mas por causa de sua facilidade em aprender, e até por isso não alcançou maiores notas. Desde cedo, se interessou por calculadoras e computadores, especialmente uma calculadora que era de seu avô. Leo Waldemar Törnqvist foi professor de estatística na Universidade de Helsinque e foi o responsável pelo seu primeiro contato com um computador, quando ele tinha 11

anos de idade. O nome desse computador era Commodore VIC-20. Segundo Torvalds, “o VIC-20 foi um dos primeiros computadores prontos feitos para a casa. Não requeria montagem” (TORVALDS; DIAMOND, 2001, p. 7). Era como um eletrodoméstico, tudo que precisava era conectá-lo à TV e ligá-lo. Porém, não havia muito que fazer com ele no início. Então, seu avô começou a programá-lo em BASIC e a partir daí pôde fazer em casa cálculos complexos que, até então, só podia fazer nos grandes computadores da universidade.

Esse contado com o computador de seu avô abriu um mundo de possibilidades a Torvalds, que começou a aprender a linguagem de programação BASIC e a criar seus próprios programas. Inicialmente, ele começou com um programa básico, que escrevia a palavra *hello* na tela, posteriormente ele desenvolveu outra versão, já um pouco mais sofisticada, pois repetia diversas vezes na tela a frase: “Sara é o máximo”, fato que causou grande impressão em sua irmã (TORVALDS; DIAMOND, 2001).

Esse enorme interesse de Torvalds pelo computador deve-se às possibilidades quase infinitas que essa máquina pôde proporcionar. Para o finlandês, computação e matemática são parecidas, pois nesses dois ambientes é possível criar um mundo próprio e através dele criar regras próprias (ISAACSON, 2014). Enquanto a maioria das crianças estavam jogando futebol ou esquiando com os pais, o jovem Linus estava no quarto de seu avô brincando com o computador. Ele achava o Commodore VIC-20 mais interessante do que as bincadeiras de criança, pois encontrava nele um universo onde a lógica governanava e onde tudo era possível (TORVALDS; DIAMOND, 2001).

Durante os anos em que Torvalds esteve na escola, seus interesses estavam voltados principalmente para matemática e física. Segundo ele próprio, essas disciplinas eram interessantes e por isso as consideravam fáceis (TORVALDS; DIAMOND, 2001). Porém, assuntos que envolviam memorização automática faziam com que seu interesse fosse diminuindo aos poucos.

Por conseguinte, o então estudante considerava a disciplina história como chata e desinteressante. No final do ensino médio, Torvalds havia passado a maior parte do tempo em computadores ou estudando sobre eles. Havia nele um sentimento de que o único momento em que não estava sonhando com computadores era quando estava fazendo atividades que não eram voltadas para memorização de fatos e datas. As atividades mais relevanes eram, por exemplo, o

estudo de fatores econômicos que afetam um país ou a razão para as monções. Em consequência às preferências de Torvalds, suas notas nas disciplinas realcionadas à matemática e física eram as melhores, porém em discplinas como educação física e trabalhos manuais não eram muito boas. Na escola em que estudava, Norssen High School, ele pode ainda aprender latim, pelo qual se interessou bastante (TORVALDS; DIAMOND, 2001).

Em 1998, Torvalds ingressou na Universidade de Helsinki para estudar Ciências da Computação, de onde só saiu oito anos depois, com o título de mestre. Durante o primeiro ano em que esteve na universidade, ele se dedicou aos estudos e não se entusiasmou muito com projetos para computadores, se saindo muito bem, com ótimo desempenho nas disciplinas. Fato que não ocorreu depois, nos demais anos. No final do primeiro ano, o jovem universitário decidiu servir ao Exército. Torvalds se dedicou, por onze meses, aos treinamentos (ao invés de oito, como normalmente se fazia na época), pois decidiu fazer o curso de oficial. Ele deixou as Forças Armadas em maio de 1990. O programador finlandês descreveu esse tempo como difícil, pois tinha de andar em bosques e florestas carregando rolos de cabos (TORVALDS; DIAMOND, 2001).

Após retornar do Exército e voltar para a universidade, Torvalds se inscreveu em alguns cursos de verão, especialmente o curso sobre Unix, pois ele havia lido sobre Unix no livro *Sistemas operacionais: design e implementação*, do professor Andrews Tanenbaum. Esse livro é lembrado por ele como o livro da sua vida. Conforme ele descreveu: “Agora todo mundo tem um livro que mudou sua vida. A Bíblia Sagrada, *Das Kapital*, *Terças com Maury*. Tanto faz. O livro que me lançou a novas alturas foi *Sistemas operacionais: design e implementação*, de Andrew S. Tanenbaum” (TORVALDS; DIAMOND, 2001, p. 51). No livro o professor discute um sistema chamado Minix, que era um pequeno clone do Unix, utilizado para estudar sistemas operacionais. Durante o curso, Torvalds se apaixonou pelo Unix, pois considerava que esse sistema era simples e eficaz. Segundo ele: “O que é especial sobre o Unix é o conjunto de ideais fundamentais pelos quais ele se esforça. É um sistema operacional limpo e bonito” (TORVALDS; DIAMOND, 2001, p. 54). O curso foi ministrado por um professor com pouca experiencia do Unix, portanto o professor e os alunos faziam o curso juntos. Apesar do professor tentar ler um capítulo a mais que o recomendado para as aulas, os alunos se adiantavam a ele e sempre faziam

perguntas para as quais ele não tinha resposta. No entanto, o curso foi um grande incentivo para Torvalds.

A forma simples como o Unix foi projetada é mencionada por ele como a causa de sua paixão. Ele descreve seis operações básicas que, para ele, resume o funcionamento do Unix e são o que fazem desse sistema o que ele é:

> Este design simples é o que me intrigou, e à maioria das pessoas, sobre Unix (bem, pelo menos nós, *geeks*). Quase tudo o que você faz no Unix é feito com apenas seis operações básicas (conhecidas por "chamadas de sistema", porque são as chamadas que você faz para o sistema operacional fazer as coisas para você). E você pode construir praticamente tudo, desde as seis chamadas básicas do sistema (TORVALDS; DIAMOND, 2001, p. 54).

Esse entendimento do Unix fez com que o cientista da computação tivesse vontade de criar um projeto de sistema operacional, partindo do Minix. Para que pudesse pôr em prática seu desejo, ele precisaria comprar um computador, o que era muito caro na época. Então, o ainda estudante Linus Torvalds juntou o dinheiro que havia ganhado no Natal e no seu aniversário e comprou um computador, pagando parte em dinheiro e financiando o restante em três anos. Ele o descreveu como uma caixa cinza e sem nome, porém tinha as especificações desejadas e, principalmente, era o que estava ao seu alcance financeiro.

O computador tinha uma versão reduzida do sistema operacional MS-DOS, porém Torvalds desejava instalar o Minix, mas este sistema não estava disponível e, por isso, foi necessário encomendá-lo. A partir daí passaram-se três meses, até que pudesse ser instalado. Com o Minix instalado, Torvalds iniciou um minucioso estudo sobre ele e descobriu que precisaria de mais programas para satisfazer suas necessidades. Por causa disso, ele passou mais de um mês adicionando seus programas favoritos com os quais estava acostumado na universidade. Mesmo assim, ainda continuava insatisfeito por causa, principalmente, do emulador de terminal. Por isso, o programador finlandês decidiu trabalhar no seu próprio programa de terminal e decidiu que esse não seria criado sob o Minix e que seu projeto seria feito diretamente no hardware. Em suas palavras: "Então, comecei um projeto para criar meu próprio programa de emulação de terminal. Eu não queria fazer o projeto sob o Minix, mas sim no nível do hardware. Este projeto de emulação de terminal seria uma grande oportunidade para aprender como o hardware 386 funcionava" (TORVALDS; DIAMOND, 2001, p. 62).

Com a escolha de escrever direto para o hardware, o cientista da computação teve de trabalhar com *assembly*, que é uma linguagem de programação legível por humanos para codificação de computadores em linguagem de máquina. Isto lhe permitiu aprender mais sobre como tirar maior proveito do próprio processador. Uma das primeiras coisas que teve de aprender foi a criar um código para leitura do teclado e apresentar os caracteres digitados na tela. Foi preciso quase um mês para conseguir uma sequência de caracteres "A" digitados e apresentados na tela. Quando conseguiu, Torvalds ficou entusiasmado, pois havia criado seu próprio programa emulador de terminal e resolveu mostrar para a irmã. Porém, o resultado não foi o esperado, como ele mesmo contou:

> […] eu estava muito orgulhoso dele. Minha irmã Sara sabia da minha grande realização pessoal. Eu mostrei para ela e ela olhou para a sequência de letras de AAAAAA e BBBBBBBB na tela por cerca de cinco segundos; então ela disse: "Bom" e foi embora, não impressionada (TORVALDS; DIAMOND, 2001, p. 63).

O emulador começou a crescer quando Torvalds decidiu que queria usá-lo para baixar coisas do computador da universidade. Para isso, precisaria de um *driver*[6] de disco que permitisse leitura e gravação no disco, além de um *driver* de sistema de arquivos para que os arquivos pudessem ser gerenciados e também pudessem ser lidos no Minix. Apesar de ser um projeto grande, o criador do Linux decidiu fazê-lo, entre outros motivos, pela falta de outros projetos mais interessantes. Durante o desenvolvimento, ele percebeu que seu terminal estava se transformando em um sistema operacional.

Após meses de trabalho e de adicionar um *shell*[7] GNU e um compilador[8] GCC, Torvalds se sentiu seguro o suficiente para publicar a versão 0.01 do Linux em um site FTP que foi disponibilizado a ele pelo assistente de ensino na Universidade de Tecnologia de Helsinque chamado Ari Lemke. Então, no dia 1º de setembro de 1991, o Linux foi disponibilizado para o mundo. No dia 25 de outubro de 1991, Linus Torvalds escreveu o e-mail que é considerado o nascimento do Linux para a comunidade:

---

[6] *Drivers* são programas que fazem a interação entre o sistema operacional e o hardware do computador.

[7] *Shell* é uma interface de usuário para acesso aos recursos do sistema operacional. É através dele que o usuário consegue enviar comandos e operar um computador.

[8] Um compilador é um programa que traduz a linguagem de programação compreensível por humanos para linguagem de máquina.

> Você suspira pelos melhores dias do Minix-1.1, quando os homens eram homens e escreviam seus próprios *driver*s para os dispositivos? Você está sem um projeto legal e morrendo de vontade de fincar os dentes em um SO que você possa modificar de acordo com suas necessidades? Você está achando frustrante quando tudo funciona no Minix? Sem ter que ficar mais a noite toda para ter um programa estiloso funcionando? Então esta mensagem pode ser exatamente para você :)
>
> Como mencionei há um mês, estou trabalhando em uma versão livre de um sistema similar ao Minix para computadores AT-386. Ele finalmente alcançou um estado usável (embora não esteja dependendo do que você quer) e estou querendo liberar as fontes para ampla distribuição. Está apenas na versão 0.02 (com +1 *patch* (muito pequeno)), mas eu executei com sucesso bash/gcc/gnu-make/gnu-sed/compress etc nele. […] Estou interessado em ouvir alguém que tenha escrito qualquer utilitário/biblioteca para o Minix. Se seus produtos forem livremente distribuídos (sob licença ou domínio público), gostaria da sua autorização, para adicioná-lo ao sistema (LINUX'S HISTORY, 2019).

Linus Torvalds é um dos gurus de informática mais venerados do mundo, considerado pai do Linux e líder de uma enorme comunidade. Porém, segundo sua própria biografia, tudo isso foi realizado apenas por diversão. Desde o início, o Linux foi distribuído de forma livre para ser copiado e distribuído gratuitamente. Torvalds jamais cobrou pelo Linux e até por isso mesmo não está entre os multimilionários da indústria de software.

2.4 Richard Stallman

Richard Matthew Stallman é um *hacker*ativista, programador, escritor e idealizador do movimento software livre. Também é o fundador da Free Software Foundation, entidade filantrópica que é utilizada para financiar a criação, a distribuição e a divulgação de softwares livres. Richard Stallman também é um palestrante renomado que leva centenas de pessoas do mundo inteiro às suas palestras sobre software livre, liberdade intelectual e educação.

Stallman nasceu em 1953, em Manhattan, Nova York, nos Estados Unidos. Sendo filho de pais divorciados, ele se sentia deslocado e nutria um desejo de mudança. Era filho de uma professora substitua, chamada Alice Lippman, e de um ex-militar da Segunda Guerra Mundial que se tornou comerciante, chamado Daniel Stallman. Durante seu período na escola, destacou-se em matemática, mas também por sua resistência em seguir certas regras que a escola impunha aos alunos que, segundo ele, os faziam competir uns contra os outros. Estudou em escolas públicas e particulares, tendo de ser transferido diversas vezes por causa da sua rebeldia.

Em suas próprias palavras: “Eu era um problema de disciplina. […] Nunca acreditei que os adultos tivessem o direito de me dar ordens. Considerava-os como qualquer outro tipo de tirano; eles apenas tinham poder” (GROSS, 1999).

Stallman era uma criança rebelde, mas, segundo ele, não por causa da rebeldia em si, mas por causa de suas convicções. “Eu tive muitas disputas com minha mãe porque ela estava tentando me obrigar a fazer coisas que não eu aguentava” (JONES, 2006). Devido ao seu desvio de comportamento, o jovem estudante acabou em uma escola para pessoas problemáticas, o que o deixou muito envergonhado, porque ele se considerava inteligente. Stallman terminou o segundo grau em 1970.

O contato de Stallman com os computadores se deu através de manuais que lhe foram oferecidos em um acampamento de verão. Em suas palavras: “Não sei em que ano foi quando eu cheguei a ver um manual no acampamento de verão. Não sei se eu tinha nove ou dez ou onze anos. Os únicos computadores que existiam então eram grandes. Um conselheiro os trouxe” (GROSS, 1999), Mais tarde, Stallman trabalhou como assistente em um laboratório de biologia da Universidade Rockefeller e, ao mesmo tempo, foi contratado pelo IBM New York Scientific Center, onde pôde ter contato com um computador e aprendeu a programar. Conforme ele contou em uma entrevista: “Eles me deixaram entrar e programar. Então, no verão, depois do meu último ano, eles me contrataram e me fizeram escrever um programa FORTRAN, e eu consegui (terminar) esse programa em algumas semanas” (GROSS, 1999).

Em 1971, Stallman ingressou em Harvard para estudar física e matemática, “[…] onde se tornou uma lenda, mesmo entre os magos da matemática” (ISAACSON, 2014, p. 373). Ainda em Harvard, continuava antissocial e tinha dificuldade em se relacionar com os colegas. Ao ser questionado sobre que tipo de colega de quarto ele queria, respondeu: “Bem, eu preferiria um colega de apartamento invisível, inaudível e intangível” (GROSS, 1999). Depois de quatro anos em Harvard, foi trabalhar no MIT, onde encontrou uma forte cultura *hacker*, com a qual se identificou imediatamente. Era uma comunidade onde *hackers*, usuários e qualquer pessoa que quisesse contribuir ou aprender eram bem-vindos. “Se algum estranho quisesse se envolver e quisesse trabalhar com o software ou trabalhar no sistema, qualquer pessoa que fosse competente seria bem-vinda” (GROSS, 1999).

Com o passar do tempo, essa comunidade foi se esvaziando, até que Stallman passou a ser o único *hacker* no MIT. Nesse período, ele se viu no meio de uma guerra de interesses entre duas empresas que estavam implantando software proprietário no MIT. Com isso, o programador passou a ser pressionado a utilizar apenas o sistema de uma das empresas. Conforme ele contou: “Eles disseram, de agora em diante, se você quiser usar qualquer um dos nossos códigos, quaisquer melhorias que estamos fazendo, você deve abandonar a versão MIT do sistema e, essencialmente, juntar-se ao nosso campo” (GROSS, 1999). Até então, Stallman se mantinha neutro, porém ele se sentiu afrontado e decidiu tomar partido contra a empresa Symbolics e passou a escrever código que fosse compatível com os dois sistemas e a deixar o código-fonte disponível.

O *hacker*ativista sabia que não poderia vencer a guerra, porém era possível diminuir a vantagem da Symbolics, fazendo assim com que seus lucros diminuíssem. Em suas próprias palavras, sua participação nesses eventos foram definidos da seguinte forma: “Esta era a Bélgica. Os alemães atacam a Bélgica, o Exército belga luta do lado dos ingleses e dos franceses” (GROSS, 1999).

Após essas experiências, Stallman começou a pensar em como reconstruir a comunidade que havia sido desfeita e que ele tanto amava. Assim, ele iniciou um novo projeto:

> […] eu senti que não queria passar o resto da minha vida punindo um ato de agressão. Eles destruíram a comunidade que eu amava. Posso construir uma nova? Então eu olhei para o que eu poderia fazer. Então, durante a primavera ou verão de 1983, comecei a procurar e percebi que, se desenvolvesse um sistema operacional livre para computadores modernos, ou seja, não especializados, […] computadores comuns, então eu poderia criar uma nova comunidade em que as pessoas poderiam compartilhar (GROSS, 1999).

Nesse período, Stallman iniciou o movimento que viria a revolucionar a computação. Ele passou a arregimentar colaboradores para formar uma nova comunidade de desenvolvedores e *hackers* para que pudesse reavivar o antigo espírito de camaradagem do MIT. Em suas próprias palavras: “Comecei a convidar pessoas para se juntarem e se ajudarem. Mas eu não comecei uma empresa. Eu comecei um projeto, um movimento” (GROSS, 1999).

Para poder seguir adiante com suas ideias, ele precisou se demitir do MIT. Isso se deu por causa de possíveis problemas com direitos autorais, pois, como

funcionário da universidade, tudo que ele criasse seria propriedade dela. No entanto, a direção do MIT permitiu que Stallman continuasse a utilizar o laboratório de inteligência artificial. Ele estava certo de que, mesmo que o MIT não permitisse sua permanência no laboratório, acabaria encontrando outro lugar para seguir com seus planos:

> Eles me deixaram continuar usando suas máquinas depois que eu parei. Eu não ia comprar um computador. Se eu não pudesse usar os computadores do Laboratório IA, tenho certeza de que encontraria alguém que me deixaria usar um computador. Eu tinha bastante reputação mesmo naquela época, se aparecesse em um departamento de ciência da computação em todo o país, provavelmente haveria alguém lá que soubesse quem eu era e diria: “Ah, você quer entrar? Venha aqui” (GROSS, 1999).

Dando sequência à sua decisão, Stallman iniciou o projeto de criação de um sistema completo para operação de computadores pessoais e criou um movimento voltado para difusão da ideia de liberdade de uso de software e uma fundação para administrar esse movimento. Segundo ele:

> Então eu comecei um movimento. Para criar uma nova comunidade. Porque eu perdi minha comunidade. Usuários, desenvolvedores, seja quem for. Porque qualquer um é bem-vindo para ser um desenvolvedor. Assim como no Laboratório de IA, se as pessoas aparecessem na rede ou visitassem pessoalmente e começassem a andar por aí e gostassem de nossas máquinas e aprendessem a programar e quisessem ajudar, nós as convidávamos para começar a fazer trabalhos (GROSS, 1999).

O encontro e as diferenças dos ideais de Richard Stallman, Bill Gates e Linus Torvalds moldaram o mercado de software e o modo de desenvolver programas e computadores. Cada um a seu modo ajudou a mudar a forma como o computador e, principalmente, o software era visto e desenvolvido, tornando o computador apenas um meio e o software a principal estrela do mundo da computação. As ideias geniais de Bill Gates a respeito das licenças de software transformaram o desenvolvedor em alguém que ajuda a definir as tendências na computação, tirando esse poder das mãos da indústria de fabricação de computadores.

O espírito de liberdade de Richard Stallman criou toda uma filosofia de computação e uma comunidade apaixonada, além de um novo mercado no setor de tecnologia da informação. O pioneirismo de Linus Torvalds proporcionou aos amantes de computação a capacidade de experimentar os ideais de liberdade de Stallman de forma prática. Portanto, os três cientistas da computação fazem parte

de um grupo muito maior de visionários que construíram e ainda constroem a revolução digital e que são responsáveis direta ou indiretamente pelo mundo totalmente computadorizado em que vivemos.

As questões técnicas e ideológicas e as contribuições para educação relacionadas a essa revolução comporão o objeto de estudos dos capítulos “3 Software livre e software proprietário: definições iniciais”, “4 Liberdade e controle sobre software” e “5 Contribuições do software livre para educação: educação cívica e emancipação do indivíduo”.

3 SOFTWARE LIVRE E SOFTWARE PROPRIETÁRIO: DEFINIÇÕES INICIAIS

Neste capítulo serão apresentados aspectos relacionados às definições de software livre e software proprietário, *open source* e licenciamento de software. Além disso, serão identificadas questões técnicas sobre esses tipos de softwares. Também será retratado o contexto em que se deu o desenvolvimento do software livre, destacando-se os sistemas GNU e Linux.

3.1 Software proprietário

Software ou programa é o que possibilita ao usuário realizar tarefas em seu computador. Entre essas tarefas, podem ser citados: acessar internet, criar planilhas eletrônicas, digitar textos, editar imagens etc. Ele é desenvolvido através de uma sequência de comandos que são depois transformados em uma linguagem que o computador entende. Segundo Silveira (2004, p. 6), software é: "[…] um conjunto de informações digitais escrito em uma linguagem de programação. A linguagem dos programadores também pode ser entendida como uma reunião coerente de centenas ou milhares de informações." Quando o software está concluído, ele pode ser distribuído de acordo com regras especificadas pelo seu desenvolvedor e são essas regras que vão definir se ele é ou não livre.

Software proprietário ou privado é aquele cujo detentor de seus direitos autorais não permite acesso ao código-fonte[9], não permite alterações ou melhorias nem a distribuição de cópias, a não ser por meio de licenças pagas e fornecidas por empresas parceiras e/ou pelo próprio desenvolvedor. Para Oliveira, Feres e Gonçalve (2017, p. 77), o desenvolvedor tem a opção de fechar o código-fonte. Caso ele opte por isso, "[…] o software será protegido via *copyright* e o conhecimento e a edição da tecnologia ali empregada ficam restritos ao detentor de seus direitos". Na definição de Stallman (2010, p. 82), "[…] software privado é um software desenvolvido para um usuário (geralmente uma organização ou empresa). Esse usuário o guarda ou usa e não o libera para o público com o código-fonte ou como um programa executável".

[9] Código-fonte é o código de programação do computador escrito em determinada linguagem de programação que pode ser entendida por outro programador e que, após ser processada pelo computador, se transforma em um software utilizável.

Não disponibilizar o código-fonte e não permitir modificações e compartilhamentos limita o uso desse software. Limita, ainda, a possibilidade de o usuário, ou alguém por ele designado, implementar melhorias ou remover partes maliciosas no código. Essas restrições estão relacionadas ao modo de licenciamento do software proprietário. Por exemplo, quando um *bug*[10] é descoberto em um software proprietário, o usuário não tem permissão para repará-lo. Sendo assim, ele deve notificar o desenvolvedor, geralmente uma empresa, que, na maioria das vezes, leva meses para lançar uma correção, e muitas vezes nem chega a ser lançada.

Da mesma forma, se um usuário adquire um software para determinada tarefa e deseja modificá-lo para atender outra demanda descoberta posteriormente, ele não está autorizado a fazê-lo. Nesse caso, será necessário comprar outro módulo ou outro software que faça o que ele deseja, isso quando esse módulo existe. Ele pode ainda aguardar por atualizações de versão que contemplem a sua necessidade. O software proprietário mantém o usuário dependente da empresa que o desenvolveu, pois somente esta está autorizada a fazer as alterações que por ventura ele necessite.

3.2 *Open source*

Open source é uma expressão utilizada para se referir a um software com código-fonte aberto. Isso significa que os usuários desse software podem ter acesso ao seu código de programação, podendo assim verificar qual o seu real funcionamento. No entanto, apesar do código-fonte ser aberto, nem sempre é possível modificá-lo ou redistribuir cópias modificadas ou não, pois o *open source* não precisa necessariamente ter licenças que possibilitem essas práticas. O código é aberto visando melhorias técnicas.

Os defensores do *open source* procuram apresentar esse modo de desenvolvimento como alternativa ao software proprietário por questões relacionadas à qualidade. Isso porque acreditam que o modelo proprietário limita o desenvolvimento de software. Segundo eles, essas limitações se dão principalmente pelo bloqueio de acesso ao código-fonte. Portanto, a solução seria migrar de uma

[10] *Bug* é um termo em inglês para inseto. Em computação, *bug* significa um erro no programa causado por um mau funcionamento ou erro durante a programação do código.

forma de comercialização de softwares baseados em propriedade para uma de prestação de serviços (RONZANI, 2011).

3.3 Software livre

O software livre respeita as liberdades individuais dos usuários no que diz respeito à sua utilização, adaptação e compartilhamento. Ele possibilita ao usuário escolher o que ele quer que seja executado em seu computador, permite ainda que ele entenda como um computador funciona e como fazer para modificar algo que não esteja certo ou com o qual ele não concorde.

Na definição de Stallman (2010, p. 77), um software é livre quando ele "[…] vem com permissão para que qualquer pessoa utilize, copie e/ou distribua, seja por escrito ou com modificações, gratuitamente ou por uma taxa. Em particular, isso significa que o código-fonte deve estar disponível." É através desse código-fonte que a comunidade de desenvolvedores poderá analisar o que o software realmente faz. Isso permite fazer modificações, caso necessário. Sem esse acesso, é impossível realizar essas análises e modificações.

Conceder o código-fonte significa que o desenvolvedor original deixou esse código à disposição para quem quiser estudá-lo e reproduzi-lo. É como se um criador de uma receita a deixasse à mostra para quem quisesse utilizá-la para reproduzir o prato ou mesmo criar outro a partir dele.

O fato de o código-fonte estar disponível possibilita à comunidade de desenvolvedores reaproveitarem trechos de código e programas completos para criar novos programas para solução de problemas, às vezes, completamente diferentes do propósito original do programa utilizado como base. Diversos produtos são ofertados no mercado de tecnologia da informação baseados em software livre. Esses produtos vão de softwares para controle de TVs até *firewalls* complexos, passando por sistemas operacionais para celulares.

O compartilhamento do código-fonte faz com que "[…] o desenvolvimento de software seja um processo evolutivo, onde uma pessoa copia um programa e reescreve partes dele para um novo recurso, e depois outra pessoa reescreve peças para adicionar outro recurso" (STALLMAN, 2010, p. 50). Sendo assim, o software que inicialmente atendia a determinada demanda vai aos poucos passando por aprimoramentos e adição de novas funcionalidades. Em alguns casos, o resultado

dessas modificações e acréscimos são programas com finalidades diferentes da original[11].

3.4 GNU e GPL

Para que o software seja considerado livre ele não pode ser colocado sob licenças que restrinjam os direitos dos usuários, mesmo quando ele é modificado e redistribuído. Para garantir que um software livre não seja colocado sob licenças restritivas, a Free Software Foundation elaborou uma série de regras que chamou de General Public License (GPL). A GPL (Licença Pública Geral) é uma licença que possibilita a criação e a distribuição de software livre.

A GPL é o que assegura a sobrevivência do movimento software livre, pois ela garante que, uma vez que o programa foi disponibilizado como livre, ele não pode ser apropriado por uma pessoa ou empresa e colocado sob regras restritivas, mesmo que passe por uma grande mudança e reformulação. Essa licença foi desenvolvida para o projeto GNU. GNU significa Gnu's Not Unix, que é um acrônimo recursivo, ou seja, uma abreviação que se refere a si mesma e que quer dizer que o GNU não é o Unix[12].

Segundo o idealizador do movimento software livre, Richard Stallman (2010, p. 80), "o GNU GPL (General Public License) é um conjunto específico de termos de distribuição para 'copyleftar'[13] um programa. O projeto GNU usa-o como os termos de distribuição para a maioria dos softwares GNU." Esse projeto visa à criação de um sistema operacional completo, com um núcleo que controla o hardware e uma série de aplicativos para usuários. Um sistema operacional é o que torna a utilização do computador possível para pessoas que não são programadores em linguagem de máquina. Bezerra (2010, p. 6) define sistema operacional da seguinte forma:

> O sistema operacional é uma camada de software colocada entre o hardware e software do computador, com o objetivo de facilitar as atividades

[11] Existem diversos softwares criados a partir de outros. Por exemplo, alguns dos sistemas que rodam em equipamentos de rede, como o Microtik Router SO, são baseados em GNU/Linux e tiveram muitas funcionalidades inerentes ao seu propósito adicionadas ao software original.

[12] Unix é um sistema para operação de comutadores que foi desenvolvido no final dos anos 1960.

[13] O *copyleft* é um software livre cujos termos de distribuição garantem que todas as cópias de todas as versões possuem mais ou menos os mesmos termos de distribuição. Esse termo seria o inverso do *copyright*. Foi uma brincadeira que um amigo de Stallman, chamado Don Hopkins, fez em uma carta enviada a ele sobre direito autoral e que acabou sendo usada para definir software livre (STALLMAN, 2010).

> dos desenvolvedores e usuários de software, uma vez que não precisam fazer acesso direto ao dispositivo, fornecendo uma interface mais amigável e intuitiva.

O sistema operacional GNU, que começou a ser desenvolvido na década de 1980, chegou ao início dos anos 1990 quase completo. Faltava, porém, o *kernel*,[14] uma parte importante sem o qual o sistema operacional não pode funcionar. Stallman (2010, p. 80) define o sistema operacional GNU como sendo "[…] o sistema operacional similar ao Unix, que é inteiramente software livre, que nós, no Projeto GNU, desenvolvemos desde 1984". Nessa época, a Free Software Foundation iniciou a construção do seu *kernel*, o Hurd, porém sem muito sucesso, o que abriu as portas para um *kernel* alternativo, criado por Linus Torvalds, o Linux. O casamento do GNU com o Linux foi um enorme sucesso nos anos seguintes.

Mesmo com o avanço na construção do sistema operacional da Free Software Foundation e com o lançamento da versão de teste completa do GNU com o Hurd, em 1996, o GNU/Linux passou a se destacar em grande parte do mercado de sistemas operacionais (STALLMAN, 2010). Esse destaque se deu tanto em computadores pessoais quanto em servidores. Segundo o site Operating System Market Share (2018), especializado em estatísticas de uso de softwares, em agosto de 2017 o GNU/Linux era o terceiro sistema operacional mais utilizado em PCs, atrás do Microsoft Windows e do Mac OS, o Hurd jamais se destacou. A imagem a seguir mostra o gráfico de utilização de sistemas operacionais nesse período:

[14] *Kernel* é o núcleo do sistema operacional, o qual representa a camada mais baixa de interface com o hardware, sendo responsável por gerenciar os recursos do sistema computacional em sua totalidade (UAB, 2018).

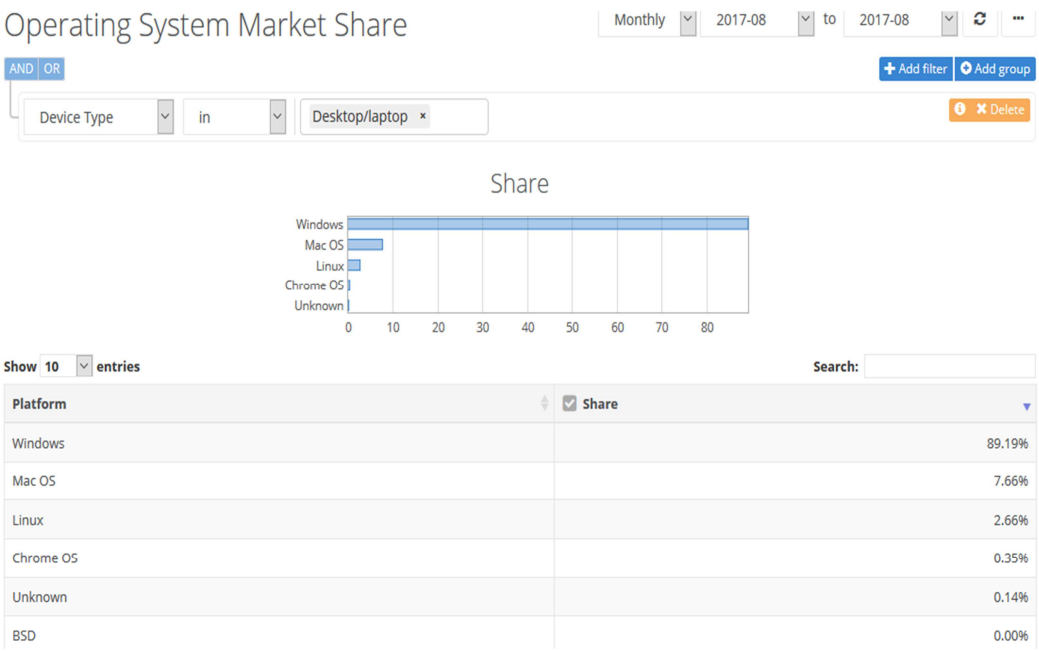

Figura 1 – Sistemas operacionais: utilização

Fonte: Imagem capturada pelo autor do site Operating System Market Share, 2018.

3.5 Desenvolvimento, eficiência, atualizações, documentação e suporte

Nesta seção serão apresentados aspectos relacionados ao desenvolvimento, à eficiência e à utilização do software livre. Além disso, serão discutidas questões práticas para os usuários e as empresas, tais como atualizações, documentação e suporte técnico.

3.5.1 Desenvolvimento de software livre

O software livre, geralmente, é desenvolvido em comunidade. Esse desenvolvimento pode se dar de várias formas, porém, normalmente, o criador do software o deixa disponível em um site na internet para que a comunidade se encarregue de auditar e aprimorar o código e para que esse código possa ser escrito com maior eficiência do ponto de vista de processamento e de segurança, além de propor melhorias também nas funcionalidades, apontando modificações ou adicionando novas. Um exemplo desse desenvolvimento comunitário é o programa Gimp, que é um software livre para edição de imagens e que possui uma seção em seu site chamada “Develop”, na qual os usuários podem colaborar com doações, reportando e resolvendo problemas, traduzindo partes do código ou mesmo

desenvolvendo funcionalidades. A figura apresentada a seguir mostra a página inicial do site de administração do software Gimp.

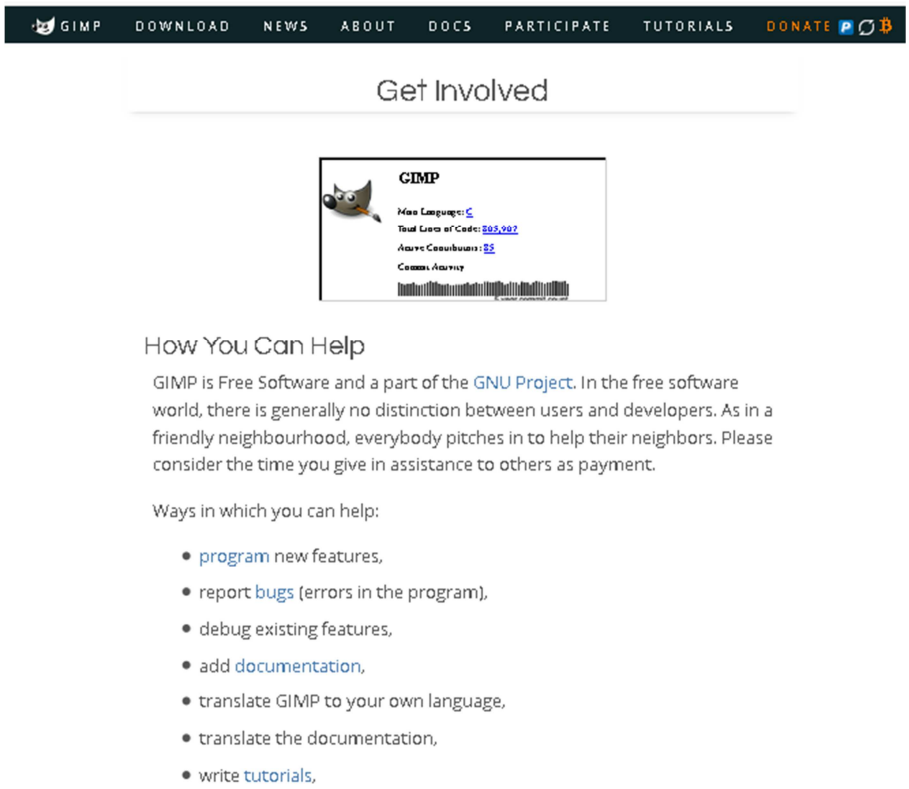

Figura 2 – Página inicial do site de administração do software Gimp

Fonte: Imagem capturada pelo autor do site Gimp (2019).

Esse desenvolvimento comunitário é uma das grandes diferenças entre software proprietário e software livre: enquanto o software proprietário tem um número limitado de desenvolvedores e testadores, o software livre tem teoricamente um número ilimitado. De acordo com Silveira (2004, p. 11):

> […] a diferença fundamental de desenvolvimento entre o software livre e o proprietário fica mais evidente ao se observar o modelo de desenho e confecção dos programas. As empresas de software proprietário trabalham somente com programadores contratados, assalariados ou terceirizados. Todo o desenvolvimento do software é interno à empresa. Já o modelo de código aberto é o modelo colaborativo que envolve programadores da empresa e todos aqueles interessados no desenvolvimento daquele software, inclusive voluntários espalhados pelo mundo. Por isso grande parte dos softwares livres possui sites na web para atrair desenvolvedores que trabalham coordenadamente pela rede mundial de computadores.

Além disso, problemas de mau funcionamento e segurança são rapidamente detectados e resolvidos no software livre. Segundo Isaacson (2014, p. 378 *apud* Raymond, 1999, p. 30), Eric Raymond propôs uma lei que chamou de Lei de Linus, segundo a qual, havendo olhos suficientes, todos os *bugs* são rasos. Situação

oposta ao software proprietário, que por vezes mantém seus problemas por anos seguidos sem solução. Por exemplo, o Windows XP passou anos com a famosa tela azul com despejo de memória. Esse erro era caracterizado pelo congelamento do sistema operacional e por uma sequência de informações que era apresentada em uma tela azul, neste caso somente uma reinicialização do sistema resolveria o problema.

O desenvolvimento de software livre nem sempre segue padrões e metodologias adotadas por fábricas de software. Estas metodologias servem "[…] como um apoio, um 'manual de como fazer' para o desenvolvedor de sistemas, visando, entre outras coisas, à elaboração de requisitos, maior produtividade e redução de riscos" (MOURA; MOREIRA, 2012, p. 2) No entanto, não existe uma metodologia própria para desenvolvimento de software livre. E, nesse caso, segundo Freitas (2009, p. 16), "[…] é preciso investigar algumas práticas de trabalho de diferentes comunidades envolvidas com esse tipo de software, para ser capaz de identificar as melhores características de cada uma e compor uma sugestão de metodologia de trabalho própria".

O fato de o software livre não possuir metodologia própria e não seguir necessariamente metodologias utilizadas em fábricas de software pode ser positivo no sentido de dar mais liberdade criativa ao desenvolvedor, porém existe o lado negativo, pois o software inicial pode apresentar maior quantidade e de *bugs*. Isso é compensado com a abertura do código e a auditoria da comunidade.

3.5.2 Eficiência

O software livre pode ser considerado uma alternativa mais eficiente do que o software proprietário, pois ele facilita sua adaptação para o propósito do usuário doméstico ou de uma empresa, ajustando-o às suas necessidades. Esse procedimento pode ser realizado tanto pelo usuário, caso ele domine programação, quanto por terceiro contratado por ele. Além disso, o tempo gasto com desenvolvimento e processamento é reduzido, devido à possibilidade de reaproveitamento de código. Caso um programador resolva construir um software para determinado fim, ele pode utilizar funções já escritas por outros programadores que estejam disponíveis como software livre. Por exemplo, se um programador vai

criar um programa de caixa para uma padaria, ele pode usar uma função de cálculo de preço através do peso do produto que esteja disponível em software livre.

Os desenvolvedores de software proprietário para empresas normalmente não oferecem soluções completas e ajustadas para o negócio de cada uma delas. Quando uma empresa necessita de um pacote de software, e opta por um software proprietário, ele nem sempre atende a todos os requisitos do negócio dessa empresa. Isso acontece pois os pacotes oferecidos não são personalizados. São pacotes que, por vezes, estão incompletos e não resolvem todos os problemas. Em outros casos, os pacotes vêm acompanhados de módulos desnecessários, onerando o projeto e inviabilizando a utilização desses softwares para a maioria das pequenas empresas.

Os exemplos disso são os chamados Enterprise Resource Planning (ERPs), que são sistemas complexos de gerenciamento de negócios e que vêm com diversos módulos. Segundo o site Portal ERP (2012):

> Podemos entender que o software ERP é um sistema de informática responsável por cuidar de todas as operações diárias de uma empresa, desde o faturamento até o balanço contábil, de compras a fluxo de caixa, de apuração de impostos a administração de pessoal, de inventário de estoque às contas a receber, do ponto dos funcionários a controle do maquinário da fábrica, enfim, todo o trabalho administrativo e operacional feito numa empresa.

Inicialmente, parece uma boa ideia adquirir softwares modulares, pois, dessa forma, a empresa compraria somente o que necessita, porém esses módulos são engessados e de difícil customização. Esse fato faz com que o custo com customizações seja elevado. Além disso, eles sempre são acompanhados de outros módulos que fazem parte de um pacote e não podem ser retirados, aumentando assim o preço final.

Com o software livre, as empresas passam a contar com possibilidades mais ajustadas, podendo escolher utilizar somente a parte do software que realmente resolve seus problemas ou pode adaptar o software aos requisitos necessários para o seu negócio. Caso a empresa disponha de uma equipe técnica de desenvolvedores, ela pode desenvolver, a partir de um software já disponibilizado pela comunidade, um software completo para o seu negócio. Se a empresa não possuir uma equipe própria, ela poderá contratar uma equipe ou um programador para esse desenvolvimento, tornando assim o software mais eficiente em relação ao

seu negócio. Além disso, existe a questão da segurança, pois, tendo acesso ao código-fonte, os desenvolvedores da empresa poderão verificar possíveis falhas e códigos maliciosos e removê-los. Esse argumento é corroborado por Alencar, quando ele afirma que:

> Quando conhecemos o código e nos apropriamos dele, podemos estar mais tranquilos com relação ao que “está rodando por detrás”, quais as rotinas, procedimentos que este ou aquele software está realizando que nos escapa aos olhos. O software é transparente para nós, não há nada de oculto. Esta vantagem tecnológica está diretamente vinculada à segurança (ALENCAR, 2007, p. 72).

O mesmo ocorre com usuários domésticos quando necessitam de alguma aplicação mais simples. Eles são obrigados a comprar pacotes completos (como é o caso do MS Office), ou programas com grande quantidade de recursos que não serão utilizados por eles. Isso também pode acontecer com software livre, porém, nesse caso, o usuário pode simplesmente remover o aplicativo ou funcionalidade indesejada. Portanto, o software livre pode ser uma alternativa eficiente para atender a empresas e usuários domésticos, pois, além de serem altamente customizáveis, são muito econômicos tanto do ponto de vista financeiro quanto do ponto de vista de utilização de recursos computacionais.

3.5.3 Atualizações

Outro ponto a ser considerado quando se fala em software são as atualizações de versão. Atualizações são pacotes que modificam parte do software com a intenção de resolver problemas na versão atual ou implementar melhorias ou novas funcionalidades. Além das atualizações que tem por finalidade resolver problemas, existem as que modificam totalmente o software. Nesse caso, o software passa para uma nova versão. Empresas de software proprietário normalmente lançam versões modificadas dos seus softwares que obrigam os seus clientes a fazerem a atualização, pois deixam de suportar a versão atual, passando então a não disponibilizar atualizações que resolvem os problemas dessa versão. Em julho de 2015, a Microsoft descontinuou o Windows Server 2003 e deixou de liberar atualizações de segurança para ele, forçando seus usuários a migrarem para novas versões (MICROSOFT, 2015). Em alguns casos, essa atualização é simples e gratuita. Em outros, porém, pode ser complexa e dispendiosa. Além de envolverem

questões de licenciamento, essas atualizações podem exigir mais do hardware, o que pode significar a troca dos equipamentos. Outras atualizações fazem com que os usuários tenham de aceitar modificações que restringem o uso do programa ou que fazem com que dados pessoais do usuário estejam disponíveis para a empresa que desenvolveu o software (STALLMAN, 2010).

As atualizações também existem em software livre e são importantes para garantir a evolução dos softwares e a resolução de problemas das versões anteriores. A diferença é que, nesse caso, as atualizações não são obrigatórias, ficando a cargo do usuário ou equipe de desenvolvedores por ele designada a decisão por atualizar ou não o sistema. Isso faz com que grande quantidade de esforço e dinheiro seja economizada, caso a atualização não seja necessária. E mesmo quando a atualização é necessária o usuário pode ainda optar por fazê-la de forma controlada, pois ele sabe exatamente o que a atualização altera no código do programa. Uma atualização que informe que vai alterar determinado trecho de código que está causando instabilidade de uma ou mais funcionalidade poderá ser verificada diretamente no código-fonte pelo usuário ou por algum programador contratado.

As atualizações em software livre são disponibilizadas pela comunidade e são amplamente testadas, fazendo com que o usuário possa ter mais segurança em aplicá-las, tendo em vista que possíveis problemas decorrentes dessa atualização são reportados por outros usuários da comunidade. Esses problemas são corrigidos e novas versões são disponibilizadas com documentação atualizada. O código-fonte da nova versão também é disponibilizado para que a comunidade possa confirmar as modificações.

3.5.4 Documentação e treinamento

Um fator importante a ser considerado em qualquer tipo de software é a mão de obra especializada, tanto para sua utilização quanto para seu suporte técnico. No passado, o software livre carecia de mão de obra e de suporte. De acordo com Garcia *et al*. (2011, p. 110), os usuários de software livre ainda não atingiram a massa crítica, assim como ocorreu com o Windows. Porém, “[…] com o crescimento do software livre, o número de profissionais qualificados tende a aumentar […]”, e assim tanto universidades quanto os cursos de capacitação estão oferecendo

treinamento especializado em software livre. Por exemplo, a universidade FUMEC oferece uma disciplina chamada Introdução aos Sistemas Operacionais *Open Source* na grade do curso de redes de computadores (FUMEC, 2017). Outro exemplo é o curso de MBA em Administração de Rede Linux da Universidade de Araraquara. Nesse curso, "[…] o aluno irá entender como funciona a infraestrutura de rede bem como as recomendações de configuração dos serviços que rodam na plataforma Linux" (UNIARA, 2018).

Grande parte das informações a respeito de software livre está disponível em livros, revistas e sites na internet, tornando-o assim acessível para quem queira aprender, independentemente de cursos ou treinamentos. Além disso, é possível encontrar diversos cursos gratuitos sobre os softwares mais populares. Esses cursos estão disponíveis tanto em apostilas quanto em videoaulas e outros formatos. A própria comunidade de software livre produz muita documentação sobre os softwares desenvolvidos, o que facilita o entendimento do funcionamento deles. Normalmente, um software é disponibilizado com seu código-fonte e a documentação relativa a ele. Segundo Stallman (2010, p. 193), "[…] o software livre precisa de documentação livre: um software livre deve vir com manuais que oferecem as mesmas liberdades que o software".

A utilização de software livre ainda é difícil para os usuários finais, o que acaba se tornando um fator negativo. Isso acontece porque a maioria dos usuários está acostumada com os softwares proprietários mais conhecidos, como é o caso do Pacote Office da Microsoft, entre outros. Muitos usuários ainda têm certa resistência em aprender a utilizar o software livre devido às dificuldades que aprender uma nova ferramenta acarreta e também pela desinformação. Silva (2012, p. 7) afirma que "[…] estes aplicativos nunca tiveram uma fácil aceitação entre os usuários, acorrendo até mesmo rejeição". Contudo, com o a evolução de softwares como o OpenOffice, Gimp, entre outros, esse cenário tende a melhorar, pois os usuários irão se familiarizar com eles.

3.5.5 Suporte

A disseminação do software livre criou uma demanda cada vez mais crescente de customização de softwares, o que gerou também a necessidade de suporte técnico para esses softwares. Dados do Instituto Brasileiro de Geografia e

Estatística (IBGE), divulgados em 2012, revelaram que 48,3% das microempresas brasileiras usaram softwares livres em 2010 (G1, 2012). Em decorrência disso, empreendedores perceberam essa demanda e criaram diversas empresas especializadas de suporte em software livre. Uma dessas empresas é a Linux Place, empresa de Belo Horizonte voltada para suporte a empresas que utilizam software livre. Essa empresa foi criada em 1999 durante o início da expansão do software livre para prestação de serviço a usuário e empresas (LINUX PLACE, 2008).

Diversos produtos de suporte são ofertados tanto por empresas quanto por consultores, barateando os custos com a assistência e a mão de obra. Algumas dessas empresas estão focadas em dar auxílio a determinados softwares conhecidos. Entretanto, outras prestam serviço de consultoria e desenvolvimento personalizados baseados em software livre em geral. Em muitos casos, essas empresas adaptam certos softwares livres ao negócio da empresa contratante, customizando e vendendo contratos de apoio. Isso é possível graças ao código-fonte aberto, pois, segundo Stallman (2010, p. 31), "[…] se sua empresa precisa poder contar com suporte, a única maneira é ter todas as fontes e ferramentas necessárias. Então você pode contratar qualquer pessoa disponível para consertar seu problema. Você não está à mercê de nenhum indivíduo."

Esse modelo de acordo baseado em software livre acrescenta qualidade e economia para as empresas, mas também é um bom negócio para os profissionais, pois abre um mercado amplo de possibilidades. Para Tavares (2007, p. 66):

> […] ao contrário do que se pensa, o SL não implica renúncia à apropriação privada, que pode ocorrer tanto pelo desenvolvimento de modelos de 1negócios voltados para serviços, desenvolvimento, treinamento, customização, habilitação de hardware, entre outros, como pela aprendizagem que possibilita a apropriação de novos conhecimentos.

O modelo de negócios e desenvolvimento do software livre é muito diferente do software proprietário, pois possuem objetivos distintos. De acordo com Silveira (2004, p. 65):

> O modelo de negócios do software livre é baseado em serviços. O modelo de software proprietário é centrado em licenças de propriedade. O primeiro, por expor seu código-fonte, busca vender desenvolvimento, capacitação e suporte especializado. O segundo vive do aprisionamento dos seus clientes ao pagamento de licenças de uso. O primeiro modelo exige que a empresa inove permanentemente para manter sua clientela. Já o segundo utiliza a vantagem das enormes dificuldades de mudança de sua solução fechada.

A prestação de serviço é a base para a maioria dos negócios em software livre. Portanto, desenvolver e dar suporte ao software livre são atividades que, além de ser importantes do ponto de vista social, podem ser também muito rentáveis tanto para empresas quanto para profissionais autônomos, utilizando-se de modelos de acordos inovadores e também tradicionais.

3.5.6 Computação confiável

Os computadores podem ser de propósito geral ou de propósito específico. Um computador de propósito geral[15] é uma máquina capaz de realizar tarefas distintas de acordo com a programação que lhe é imposta. Aguiar (2011, p. 113) afirma que "[…] os computadores de propósito geral são projetados e desenvolvidos para que qualquer tipo de tarefa possa ser executada nele com algum desempenho mínimo garantido". Em teoria, computadores pessoais são de propósito geral. Porém, existem limitações tanto físicas quanto lógicas, adicionadas pelas empresas de computação para restringir sua utilização. Contudo, Stallman afirma que há um tipo de funcionalidade que está sendo imposta pelas empresas que controlam o mercado de tecnologia da informação mundial que está disfarçada de proteção ao usuário, conhecida como computação confiável. Ele ainda afirma o seguinte:

> Computação confiável é um plano de grandes corporações de mídia, juntamente com empresas de informática como a Microsoft e Intel. Elas estão planejando fazer seu computador obedecer a elas ao invés de você. Esse plano é possível graças ao software proprietário, pois seu código-fonte é fechado e não é possível saber o que ele realmente faz, pois não pode estudá-lo. Não é surpreendente que empresários inteligentes encontram maneiras de usar este controle para colocá-lo em desvantagem (STALLMAN, 2010, p. 205).

As restrições vão desde proibição de instalação de determinados tipos de softwares até o bloqueio de mensagens. Por exemplo, uma empresa de hardware pode bloquear a instalação de sistemas operacionais livres em computadores específicos ou uma empresa de software pode adicionar uma funcionalidade para

[15] Computadores de propósito específico realizam apenas algumas operações, enquanto computadores de propósito geral são capazes de calcular qualquer função computável. Por exemplo, um computador de mesa ou Desktop pode ser programado para diversas funções, como executar planilhas eletrônicas, editar imagens, calcular trajetórias balísticas, exibir vídeos. Já calculadoras eletrônicas executam apenas cálculos específicos definidos pelo fabricante e não são programáveis pelo usuário, outro exemplo são os computadores de bordo dos carros que realizam somente as funções predefinidas pelo fabricante.

que os e-mails tenham validade e sejam apagados após certa data (STALLMAN, 2010). Em contrapartida, Kropiwiec (2005, p. 3 *apud* ANDERSON, 2003) afirma que o termo computação confiável se refere a "[…] uma plataforma que tem por objetivo garantir a impossibilidade de corromper uma aplicação, e que as aplicações executadas nesse ambiente possam se comunicar de forma segura entre si e entre outras plataformas de computação confiável". Nesse sentido, segundo o autor, essa plataforma seria uma ferramenta de proteção e de garantia de confiabilidade de dados.

Contudo, Stallman (2010) insiste que esses argumentos de que colocar limitações no uso do computador protegerão o usuário de invasões e ataques de *crackers*, além de evitar a contaminação por vírus, são apenas parte de uma estratégia de controle. Segundo ele, essas empresas adicionam limitações no hardware que impedem a instalação de determinados sistemas operacionais, ou limitações lógicas nos sistemas proprietários que impedem a instalação de softwares não autorizados. Essas limitações podem vir através de criptografia, em que as chaves estão em poder do proprietário do software e só são liberadas mediante pagamento. Como acontece quando uma licença de software vence e não é renovada pelo usuário e, diante disso, o proprietário bloqueia o seu uso, até que seja feito novo pagamento. Por exemplo, o software de antivírus da Trend Micro possui, nos termos de contrato, uma cláusula que permite ao usuário a utilização do software e sua atualização enquanto durar a vigência do contrato:

> […] durante um ano a partir da data na qual o usuário recebeu o número de série, a chave de registro ou o código de ativação, ou a confirmação do pedido do produto, o que ocorrer primeiro ("Prazo de Manutenção"). A fim de preservar os direitos de manutenção, o usuário deverá adquirir uma manutenção com renovação anual de seu fornecedor (ou da Trend Micro) antes de expirar o prazo de manutenção. Depois de expirado o prazo de manutenção, o usuário não terá direito à manutenção, exceto caso adquira da Trend Micro ou de um de seus revendedores autorizados uma nova renovação ou manutenção anual (TREND MICRO, 2013).

Esse tipo de licença permite que os proprietários controlem o que pode ser executado nos computadores dos usuários. Limitando tanto a instalação de programas quanto o acesso a dados pessoais. Isso pode ocorrer quando determinado dado em seu computador está em um formato cujo software é proibido via computação confiável; como você não pode instalá-lo, você não terá como abrir o dado. Por exemplo, um arquivo de texto em formato diferente do ".doc" não poderá

abrir no MS Word. Esse plano é possível graças ao software proprietário, pois seu código-fonte é fechado e não é possível saber o que ele realmente faz, pois não é possível estudá-lo.

Essas afirmações são corroboradas por Anderson (2002), quando afirma que, com a computação confiável, os mecanismos de criptografia e geração de meios de comprovação de identidade permitem que dados importantes sejam criptografados, mas também permitem aos softwares instalados restringirem o acesso a essas informações a partir de softwares de terceiros que, em teoria, poderiam interoperar com eles. Além disso, segundo Stallman (2010, p. 205):

> Esses recursos maliciosos são muitas vezes secretos, mas mesmo quando você sabe sobre eles é difícil removê-los, já que você não possui o código-fonte. No passado, esses eram incidentes isolados. “Computação confiável” tornaria a prática difundida. “Computação traiçoeira” é um nome mais apropriado, porque o plano foi projetado para garantir que seu computador sistematicamente desobedeça a você. Na verdade, ele é projetado para impedir que seu computador funcione como um computador de propósito geral. Toda operação pode exigir permissão explícita.

Essa perda de controle e a impossibilidade de um computador ser de propósito geral são, para Kropiwiec (2005), fatores que desencadeiam as críticas em torno da computação confiável. Segundo o autor, as críticas se originam “[…] dos problemas com a especificação de *Trusted Computing*, angariando críticos que apontam para a supressão do controle do usuário sobre o computador, para a existência de mecanismos que poderiam levar a práticas anticompetitivas”. Essas restrições podem ser evitadas, simplesmente não utilizando softwares proprietários, em todos os casos em que isso seja possível. Pois, com software livre, o usuário é responsável pelo que seu computador faz, se você ou outra pessoa descobre alguma funcionalidade mal-intencionada, você mesmo pode corrigi-la ou informar a comunidade para que outros trabalhem na correção e a disponibilizem para o público.

Com a chamada computação confiável, existe o risco de que sistemas operacionais e softwares livres deixem de existir devido às restrições em hardware e software direcionadas a eles (STALLMAN, 2010). Já estão à venda computadores pessoais que possuem uma trava de hardware que permite somente a execução do Microsoft Windows, e, quando o usuário instala um sistema operacional livre, ele não

inicia o *boot*[16]. Segundo o site especializado em tecnologia, Techtudo (2016), em 2016 a Microsoft, juntamente com a fabricante de computadores Lenovo, comercializou um novo notebook chamado Yoga, que impossibilita o usuário de instalar sistemas GNU/Linux. Segundo o site, o problema "ocorre porque a BIOS[17] usa uma tecnologia nova que impede mudanças de configuração, fazendo com que as mídias de instalação de distribuições do Linux, como o Ubuntu, não reconheçam o SSD, o que impede seu uso".

As empresas de computação definem o que os computadores podem fazer e, portanto, definem o nível de confiabilidade que um usuário pode ter em um computador. Com isso, essas mesmas empresas tentam passar para seus usuários a ideia de confiabilidade.

3.6 Direitos autorais e Patentes de software

O desenvolvimento de software está ligado à ideia de propriedade intelectual. Essa ligação faz com que os desenvolvedores queiram requerer patentes para suas criações. Na definição do Sebrae (2017):

> Uma patente é um documento formal, expedido por uma repartição pública, por meio do qual se conferem e se reconhecem direitos de propriedade e uso exclusivo para uma invenção descrita amplamente. Trata-se de um privilégio concedido pelo Estado aos inventores (pessoas física ou jurídica) detentores do direito de invenção de produtos e processos de fabricação, ou aperfeiçoamento de algum já existente.

Nesse sentido, Silveira (2004, p. 32) afirma que:

> A patente não pode ser conferida a uma ideia ou a uma descoberta. Ela foi criada para proteger invenções originais voltadas para a utilização industrial. A proteção só é feita sobre um objeto de uso prático. Processos como planos de saúde, métodos de ensino, esquemas de descontos em lojas ou, ainda, ideias abstratas e inventos que não possam ser industrializados não são objetos de patentes. Também não são patenteáveis inventos que decorram de um uso óbvio ou comum da técnica vigente.

[16] *Boot* é um termo em inglês utilizado para fazer referência ao processo de inicialização de um computador, o qual acontece no momento em que você pressiona o botão "Ligar" da máquina até o total carregamento do sistema operacional instalado (TECMUNDO, 2012).

[17] O BIOS é um programa de computador pré-gravado em memória permanente (*firmware*) executado por um computador quando ligado. Ele é responsável pelo suporte básico de acesso ao hardware, bem como por iniciar a carga do sistema operacional (WIKIPÉDIA, 2019).

Assim, as ideias não são patenteáveis, porque não são passiveis de serem utilizadas por alguém em detrimento de outro. Isto é, mesmo que alguém utilize uma ideia, outra pessoa ainda poderá utilizá-la. Para Silveira (2004, p. 26), “[…] as ideias são bens não rivais. A criação de uma máquina, um novo software ou um novo processo, ou seja, a ideia de como fazê-los permite que vários agentes a realizem ao mesmo tempo, em distintos lugares.” Sendo assim, as ideias são diferentes de bens materiais que são rivais, ou seja, o uso de um bem por uma pessoa impede que outra a use. Por exemplo, enquanto alguém está utilizando uma chave de fenda para apertar um parafuso, outra pessoa não poderá utilizá-la para apertar outro parafuso.

No entanto, um código fonte de software não é em si patenteável. Software está protegido no Brasil pela lei 9.609 de 1998. Em seus três primeiros artigos a Lei define o que é um programa de computador, como ele pode ser protegido e como pode ser registrado.

> Art. 1º Programa de computador é a expressão de um conjunto organizado de instruções em linguagem natural ou codificada, contida em suporte físico de qualquer natureza, de emprego necessário em máquinas automáticas de tratamento da informação, dispositivos, instrumentos ou equipamentos periféricos, baseados em técnica digital ou análoga, para fazê-los funcionar de modo e para fins determinados. Art. 2º O regime de proteção à propriedade intelectual de programa de computador é o conferido às obras literárias pela legislação de direitos autorais e conexos vigentes no País, observado o disposto nesta Lei. Art. 3º Os programas de computador poderão, a critério do titular, ser registrados em órgão ou entidade a ser designado por ato do Poder Executivo, por iniciativa do Ministério responsável pela política de ciência e tecnologia. (BRASIL, 1998)

Um programa de computador está protegido desde sua concepção pela legislação de direitos autorais. Neste sentido, o criador do código possui plenos direitos sobre ele e sobre a decisão de como ele pode ser distribuído e comercializado. Este direito está garantido por cinquenta anos a contar a partir de primeiro de Janeiro do ano seguinte a sua publicação ou criação. O programador pode ainda requerer um registro em um órgão público para garantir estes direitos. Mesmo assim, ainda é possível requerer a patente de um software desde que ele atenda a requisitos de novidade, invenção e aplicação industrial definidos na legislação. Além disso, se faz necessário que este software esteja atrelado ao hardware e seja registrado como um conjunto. Por exemplo, um software embarcado em um equipamento de controle de tráfego de pessoas e seu software. (NYBO; LIPO, 2016)

A lei de direitos autorais garante que códigos fontes se tornem propriedade intelectual de quem os desenvolveu, não podendo ser usado por outros desenvolvedores livremente sem uma autorização prévia do seu proprietário. Caso outra pessoa necessite realizar uma tarefa que esse código faz, ela tem a opção de solicitar ao desenvolvedor ou criar todo o código novamente. O problema começa quando o proprietário do código é uma empresa ou um desenvolvedor que não está disposto a compartilhar ou vender esse código. Nesse caso, será preciso reescrever todo o código, o que pode levar muito tempo e dinheiro. Essa tarefa pode ter de ser realizada diversas vezes e por diversos programadores diferentes, levando a grande desperdício de tempo e dinheiro.

A dificuldade em identificar códigos que já são propriedade de alguém gera uma insegurança para os desenvolvedores de software, principalmente software livre, pois diversos componentes que são utilizados em software livre podem estar cobertos por algum registro e que só vai aparecer depois do software desenvolvido e em distribuição (STALLMAN, 2010).

A implicação técnica do enquadramento de software em leis de direitos autorais se dá principalmente pelo fato de que software geralmente é desenvolvido com ideias genéricas que poderiam ser desenvolvidas por qualquer programador, sem que o mesmo soubesse que se trata de algo protegido por direitos autorais. Em 1984, um algoritmo de compressão foi utilizado para criar um programa chamado Compress. Esse algoritmo não registrado por nenhum desenvolvedor e era disponibilizado em um periódico sobre informática. Porém, quando o programa foi lançado, em 1985, já havia um registro para o algoritmo. Assim, o programa Compress não poderia mais ser usado. Esse caso exemplifica o problema do registro de ideias genéricas como algoritmos.

Esse “monopólio de uma ideia” é prejudicial às pessoas que desenvolvem software, mas, acima de tudo, é prejudicial aos usuários de software. Por causa disso, diversos softwares úteis deixam de ser desenvolvidos e os problemas nos softwares existentes não são corrigidos.

3.7 GNU/Linux

Nesta seção será apresentado o *kernel* Linux, com um breve histórico de sua criação e de sua junção com o sistema GNU. Além disso, será apresentado o conceito de distribuição e sua importância para o movimento de software livre.

3.7.1 Linux

Conforme abordado no capítulo 2, o Linux é o *kernel* desenvolvido em 1991 por Linus Torvalds, um estudante da Universidade de Helsinque, na Finlândia, a partir de um projeto de um sistema operacional chamado Minix, que foi criado pelo professor Andrew Tanenbaum para o ensino de computação (ISAACSON, 2014, p. 376). Após adquirir um computador pessoal, que viera acompanhado do sistema operacional MS-DOS, Torvalds decidiu remover esse sistema de seu computador. O jovem programador havia se decepcionado com ele e queria algo mais parecido com o sistema operacional Unix, que era para computadores de grande porte e que foi o padrão de mercado por muitos anos, pois era o sistema com o qual ele estava acostumado. Após ler sobre o Minix no livro *Sistemas operacionais: projeto e implementação*, de Tanenbaum, Torvalds decidiu que iria substituir o MS-DOS pelo Minix. Para adquirir o sistema, foi preciso pagar uma taxa de 69 dólares. Mais tarde, ele disse que achou um absurdo o preço. Após finalizar a instalação e depois de realizar algumas customizações, ele começou a trabalhar em seu sistema (ISAACSON, 2014, p. 376).

Depois de escrever um programa para emular um terminal onde ele pudesse digitar os comandos para o computador e alguns *drivers* para leitura de discos e sistema de arquivos, ele se deu conta de que estava com um projeto de sistema operacional. Durante o desenvolvimento do Linux, Torvalds resolveu liberá-lo como software livre para que a comunidade pudesse ajudá-lo a aprimorar o código. Isaacson (2014, p. 376) afirma que:

> Em vez de tentar comercializar o que havia produzido, ele decidiu simplesmente oferecê-lo ao público. […] ele viu as vantagens práticas da abordagem aberta. Quase por instinto, e não como escolha filosófica, achou que o Linux deveria ser compartilhado livremente, com a esperança de que quem o usasse pudesse ajudar a aperfeiçoá-lo.

Torvalds (TORVALDS; DIAMOND, 2001, p. 88) disse que uma das coisas que o fez distribuir o Linux foi para

> provar que não era apenas ar quente, que eu tinha realmente feito alguma coisa. Na Internet, falar é fácil. Independentemente do que você faz, sejam sistemas operacionais ou sexo, muitas pessoas estão fingindo no ciberespaço. Então é bom, depois de conversar com muitas pessoas sobre a construção de um sistema operacional, ser capaz de dizer, “Veja, eu realmente tenho algo feito. Eu não estava te enrolando. Está aqui o que eu tenho feito.”

Inicialmente, Torvalds decidiu utilizar o nome Freax. Mas administradores do site FTP, onde os arquivos estavam sendo disponibilizados, não gostaram do nome. Por isso, Torvalds liberou a versão inicial com o nome de Linux. O nome Linux é a junção do nome Linus com Unix, conforme Torvalds afirmou no Livro *Só de brincadeira: a história de um revolucionário acidental* (TORVALDS; DIAMOND, 2001, p. 84):

> […] eu estava compilando tudo no Minix, mas movi o *shell* para uma partição especial que eu tinha criado para o novo sistema operacional. Privadamente eu chamei Linux. Honestamente: Eu não queria liberá-lo sob o nome de Linux porque era muito egoísta. Qual era o nome que eu reservava para qualquer eventual lançamento? Freax.

A liberação do código impulsionou o desenvolvimento do Linux, que, aos poucos, foi ganhando funcionalidades e tendo seu código melhorado, através de remoção de erros de funcionamento. Por exemplo, era comum em sistemas Linux um erro conhecido como *kernel panic*. Esse erro está relacionado ao acesso do *kernel* ao hardware. Porém, com o desenvolvimento tanto do *kernel* quanto dos programas dos *hardwares*, aquele erro passou a acontecer com menos frequência. No entanto, o sucesso do Linux não decorreu somente do *kernel.* Pacotes de software aplicativos e das interfaces gráficas também foram adicionados a ele, como a interface gráfica Gnome e os softwares Open Office, Gimp, Gzip, entre outros. A maioria dos softwares vinham do projeto GNU e da Free Software Foundation, que já havia criado dezenas de softwares que foram muito úteis no desenvolvimento do Linux. A partir da união dos softwares do projeto GNU com o *kernel* Linux, surgiu o GNU/Linux.

3.7.2 Distribuições

Com a possibilidade de adição de softwares aplicativos ao sistema GNU juntamente com o *kernel* Linux, tanto usuários quanto comunidades e empresas puderam criar pacotes de softwares personalizados. Esses pacotes são conhecidos como distribuições ou “distros”. Conforme afirma Silveira (2004, p. 65):

> A cada ano o coletivo mundial de desenvolvedores coordenados por um mantenedor central lança novas versões deste *kernel*. Também chamadas de *releases*, estas versões são empacotadas de maneiras diferentes por vários coletivos comerciais e comunitários, que são chamamos de distribuições ou simplesmente “distros”.

Essas distribuições contêm tudo que um sistema operacional necessita para funcionar e, além disso, são acompanhadas de diversos softwares aplicativos. Na maioria das distribuições, é adicionada ainda uma GUI[18], uma interface gráfica que possibilita ao usuário a utilização do computador através de mouse e de janelas. Ela facilita a utilização do GNU/Linux por usuários com menor conhecimento em computação. Isso fez com que o GNU/Linux se tornasse mais competitivo em relação ao Microsoft Windows no mercado de sistemas operacionais para computadores pessoais.

As distribuições são importantes pelo fato de grupos ou comunidades poderem criar suas próprias customizações e empacotá-las e as distribuir. Elas ajudam usuários com pouco conhecimento a utilizar ferramentas específicas, sem ter de contratarem profissionais para fazerem customizações extras.

Existem distribuições voltadas para os mais diversos fins, como, por exemplo, servidores de aplicações e web, educacionais, games, *hackers*, escritório, usuários domésticos, aplicações científicas, desenvolvedores etc. Como exemplo, é possível citar o Kali Linux, voltado principalmente para auditoria e segurança de computadores em geral, o Edubuntu, voltado para ambientes escolares, o Scientific Linux, distribuição desenvolvida para laboratórios científicos, entre outros. Cada qual com especificidades e com ferramentas próprias para o público a que se destinam.

[18] Uma Interface Gráfica de Usuário, ou GUI, é, basicamente, um software que oferece e realiza a interação entre o usuário final e a máquina em si. A GUI é responsável por exibir e gerenciar janelas de programas e apresentar ao usuário final uma forma prática e amigável de utilizar o computador sem a necessidade de aprender uma série de comandos e linhas de código para poder realizar suas tarefas (BLOG TIQX, 2015).

Dessa forma, ferramentas que não serão úteis para a comunidade a que se destina a distribuição não são adicionadas ao pacote.

Algumas empresas, como a Red Hat, mantém uma distribuição comercial para ser disponibilizada com suporte e atualização e mantém, ao mesmo tempo, outra para comunidade que, no caso da Red Hat, é o CentOS. Porém, nenhuma das duas é considerada software livre pela Free Software Foundation, pois possui em seu pacote de software código-fonte proprietário.

Em alguns casos, as distribuições incorporam em seu pacote de softwares aplicativos desenvolvidos sob licenças proprietárias. Esses podem ser gratuitos ou que são de teste gratuito por um período, mas que não fornecem o código-fonte. Sendo assim, essas distribuições deixam de ser totalmente livres. Isso ocorre porque essas distribuições são desenvolvidas por vários grupos diferentes que nem sempre estão comprometidos com a filosofia do software livre.

A Free Software Foundation possui critérios para definição de uma distribuição como livre ou não livre, conforme a seguinte diretriz:

> A distribuição de um sistema livre não deve levar os usuários à obtenção de qualquer informação não livre para uso prático ou encorajá-los a fazê-lo. O sistema não deve possuir nenhum repositório para software não livre e nenhuma instrução específica para instalação de programas não livres em particular. A distribuição também não deve fazer referência a repositórios de terceiros que não estão comprometidos a incluir somente software livre; mesmo se eles possuírem somente software livre hoje, isso pode não ser verdade amanhã. Programas presentes no sistema não devem sugerir a instalação de *plug-ins*, documentação e outros componentes não livres (FREE SOFTWARE FOUNDATION, 2018).

Esses critérios visam garantir que as distribuições sejam realmente livres e que não induzam o usuário a utilizar software proprietário pensando estar usando software livre. Para o movimento de software livre, as distribuições são importantes para sua difusão e para aproximar os usuários dele. No entanto, essas distribuições devem seguir os critérios definidos pela Free Software Foundation para que elas sejam um instrumento a favor desse movimento.

4 LIBERDADE E CONTROLE SOBRE SOFTWARE

Este capítulo teve como objetivo apresentar os aspectos sociais e filosóficos contidos no movimento software livre e nas estratégias das empresas de software proprietário. Esses aspectos se refletem na construção do software e nas suas questões técnicas, pois as funcionalidades contidas neles são o resultado de definições políticas feitas por quem os idealizou.

4.1 Sucesso do movimento software livre de Richard Stallman

O movimento software livre tem aspectos relacionados ao ativismo social[19] em defesa dos direitos e das liberdades individuais, tais como liberdade de expressão, de escolha e de pensamento. Esse movimento atua de forma política e também prática, trabalhando questões técnicas para que elas auxiliem na defesa das liberdades e da disseminação de conhecimento. Segundo Gomes, Novaes e BECKE (2016, p. 313):

> A referência aos softwares como livres alude à sua ideologia de respeito às liberdades essenciais que são fundamentais para a sociedade, principalmente ao se considerar a digitalização de nossas atividades diárias, que dependem do bom desempenho dos softwares que as estruturam. Nesse contexto a criação e a divulgação de softwares livres tem o poder de dar acesso tecnológico relevante a diversas sociedades, promovendo o desenvolvimento econômico e a liberdade em geral.

Para que esse movimento tenha sucesso, é importante que tanto desenvolvedores quanto usuários de sistemas de computadores sejam conscientizados sobre a importância dessas liberdades. Porém, existe um grande obstáculo na conscientização desses usuários. Muitos deles utilizam o sistema GNU/Linux, porém acreditam que estão usando sistema Linux e com isso carregam para si a visão de mundo e de liberdade do criador do Linux. Segundo Stallman (2013), os usuários de GNU/Linux foram levados a acreditar que toda luta por software livre começou em 1992, com Linus Torvalds e com o *kernel* Linux. Assim, passaram a admirá-lo e a seguir seus ideais. Esse erro ocorreu porque parte da

[19] O ativismo social está ligado a trabalhos sociais. O ativista social e tido com um agente de transformação da sociedade.

comunidade começou a chamar o sistema por Linux; e, mais tarde, os usuários e, por fim, os meios de comunicação passaram a imitá-los.

Os ideais e a visão de mundo do criador do Linux (Linus Torvalds) não são os mesmos do movimento software livre, pois, para ele, a liberdade na utilização de software não é o mais importante, mas sim a eficiência, que deve ser alcançada em primeiro lugar. Isaacson (2014, p. 378) afirma que, quando Torvalds decidiu liberar o Linux como software livre, sua intenção era aprimorar o software que ele havia criado. Portanto:

> Torvalds decidiu usar a Licença Pública Geral do GNU não porque concordasse totalmente com a ideologia de compartilhamento livre de Stallman (e também de seus pais), mas porque achava que deixar os hackers do mundo todo pôr as mãos no código-fonte levaria a um esforço colaborativo aberto que resultaria num software mais impressionante.

Stallman (2013) explica que, depois de vários anos de luta para se conseguir um sistema operacional totalmente livre, em 1992 esse sistema ficou pronto com a inclusão do *kernel* Linux ao sistema GNU. Com isso, os usuários de software conseguiram atingir, enfim, a liberdade. Ainda segundo Stallman, após atingir a liberdade, a comunidade não a valorizou, deixando-se influenciar pelos ideais de Torvalds. A partir daí passaram a adicionar funcionalidades com código proprietário na maioria das distribuições e até no próprio *kernel*. Assim, o sistema Linux deixou de ser totalmente livre. Pouco tempo depois, quase nenhuma distribuição era totalmente livre. Portanto, será preciso que se faça uma distinção clara do projeto GNU e do *kernel* Linux para que seus usuários não sejam levados a acreditar que os ideais de seus criadores são os mesmos.

Outro importante passo para o sucesso do movimento software livre é distinguir o software livre do software proprietário em termos sociais. Stallman (2013) afirma que não se deve distinguir software livre do software proprietário apenas tecnicamente, pois, segundo ele, o que importa nesse caso não é somente o funcionamento do código-fonte nem o trabalho envolvido em seu desenvolvimento. A distinção que se deve ter em mente é a distinção ética, a distinção social e política, pois se trata de um sistema social para os usuários desse software.

Na visão de Stallman (2013), o software livre é desenvolvido democraticamente com a participação da sociedade e pensando na comunidade e nos interesses dos usuários. Portanto, desenvolvê-lo contribui para a democracia na

medida em que disponibiliza a todos os seus usuários a possibilidade de decidir qual uso seu computador terá. Além disso, esse software também disponibiliza aos usuários o livre acesso ao conhecimento do funcionamento de um software. Em oposição a isso, o software proprietário é desenvolvido sob a ditadura das grandes corporações e dos interesses individuais, buscando somente o controle e o lucro.

O autor afirma também que utilizar software livre em uma sociedade pode contribuir para o desenvolvimento social, pois, nesse caso, mais informação estará disponível para todos. Desse ponto de vista, o uso de software privado poderá ser considerado dependência e não desenvolvimento social, pois faz com que seus usuários dependam do autor do software para conseguir mais informações.

Nesse caso, o uso de software privado causará um problema social. Por isso as comunidades devem tentar eliminar esse problema passando a utilizar software livre ao invés de software privado.

4.2 Divulgação e disseminação do software livre

Os criadores do movimento software livre estão em constante luta para difundir seus ideais e para popularizar a filosofia do software livre. Durante essa luta, eles se deparam com ferozes opositores. Entre eles, estão empresas multimilionárias como Microsoft, IBM, Apple, Amazon, Google e Facebook. Portanto, tal luta é travada diariamente. Para isso, esse movimento conta com milhares de colaboradores espalhados pelo mundo que, munidos de boa vontade, espírito comunitário e conhecimento em computação, põem em prática o desenvolvimento de diversos softwares livres. Silveira (2004, p. 14) afirma que “[…] existem diversos softwares livres que possuem comunidades de desenvolvedores espalhados por diversos países e com milhares de colaboradores que auxiliam na sua constante melhoria e correção”. Grande parte destes colaboradores atuam também como multiplicadores dos ideais e da filosofia do software livre.

O software livre e seus defensores têm conseguido cada vez mais espaço para divulgação, tanto na mídia quanto em universidades, que cedem seus auditórios para palestras, que geralmente ficam lotadas de entusiastas e estudantes. Por exemplo, em maio de 2017, Richard Stallman proferiu a conferência “A Free Digital Society: What Makes Digital Inclusion Good or Bad?” (Uma sociedade digital gratuita: o que torna a inclusão digital boa ou ruim?), durante as comemorações de

90 anos da UFMG. Foram necessários dois auditórios para acomodar o grande número de pessoas que compareceram ao evento. Porém, mesmo com toda essa popularidade, ainda existe grande dificuldade em relação à concorrência com a divulgação dos softwares proprietários, pois esses contam com campanhas publicitárias milionárias. Segundo o jornal *The New York Times*, as empresas Microsoft e Apple entraram em uma disputa midiática. Em 2008, a Apple gastou US$ 264 milhões em anúncios televisivos, enquanto apenas nos primeiros seis primeiros meses de 2009 a Microsoft já havia gasto US$ 163 milhões (LEONARD, 2009). Para tentar diminuir a desigualdade entre essas divulgações o *hacker*ativista, Richard Stallman e outros defensores do software livre, como Jon “Maddog” Hall, diretor do Conselho do Linux Professional Institute, e Alexandre Oliva, conselheiro da Free Software Foundation da América Latina, realizaram palestras sobre a ideologia *hacker* e sobre a filosofia do software livre. Pode-se destacar a palestra de Jon Hall sobre educação aberta: “Open Education: Do I Really Need University Education in Comp. Science and Engineering?” (Educação aberta: eu realmente preciso de educação universitária nas Ciências da Computação e Engenharia?). Além disso, diversos eventos independentes voltados para discussão e divulgação da filosofia do software ocorrem com frequência. Um desses eventos é o Fórum Internacional Software Livre (FISL) que acontece anualmente em Porto Alegre, Rio Grande do Sul, no Brasil. Esse evento proporciona “[…] uma discussão técnica, política e social sobre software livre de forma integrada. Reúne discussões, palestras, personalidades e novidades nacionais e internacionais do mundo do software livre” (FISL, 2014).

No entanto, faz-se necessário um esforço conjunto tanto das comunidades de software livre quanto de usuários, programadores e instituições de ensino para que o software livre seja divulgado e conhecido pelo maior número possível de pessoas. A disseminação do software livre poderá levar à disseminação do conhecimento em geral, no sentido de democratizar o acesso à informação. Nessa linha de pensamento, Silveira (2004, p. 7) afirma que “[…] a transmissão e a disseminação do conhecimento tecnológico permitem viabilizar o fortalecimento da inteligência coletiva local e evitar a submissão e o aprisionamento pela inteligência monopolista”.

4.3 Controle do desenvolvimento de software

Atualmente, o controle sobre o desenvolvimento e distribuição de software está principalmente sob domínio de empresas. Essas empresas que desenvolvem software com empresas que fabricam o hardware para os computadores decidem o que o usuário pode processar e armazenar em seu computador/celular, limitando assim o uso deles. Por exemplo, a Apple só permite instalar em seus dispositivos aplicativos homologados e disponibilizados em sua loja virtual. Aplicativos desenvolvidos para Android ou outra plataforma e que tenham versões adaptadas para iPhone, mas que não foram aprovados na loja virtual da Apple, não podem ser instalados, sob pena de perda de garantia do aparelho. Existem alternativas para contornar essa limitação, porém são consideradas ilegais, e a própria Apple denomina essas alternativas como *jailbreak*, algo como “fuga de presos”. Admitindo assim que seus usuários estão presos em sua loja virtual (STALLMAN, 2013)

Esse controle possibilita que a empresa proprietária de um software possa adicionar funcionalidades que são diferentes das originais para o qual o software foi desenvolvido. Alguns softwares proprietários possuem funcionalidades conhecidas como *backdoor*, ou “porta dos fundos”. São funcionalidades que permitem às empresas proprietárias controlar esses softwares remotamente e, nesse caso, controlar o computador ou celular do usuário (STALLMAN, 2013). Isso permite que elas modifiquem, instalem, removam ou copiem qualquer programa ou arquivo desses computadores. Segundo o site Software Livre Brasil (SOFTWARELIVRE, 2013), o Windows possui uma dessas portas desde 1999 para que a Agência de Segurança Nacional (NSA) dos Estados Unidos pudesse ter acesso a dados de usuário. Outro exemplo é o caso em que a Amazon, em 2009, apagou livros que os usuários haviam comprado e estavam armazenados no leitor de livros eletrônicos chamado Kindle. Curiosamente, o livro era *1984*, do famoso escritor George Orwell (VENTURA, 2016).

Além disso, elas podem instalar software remotamente, sem o consentimento do usuário. Em um caso admitido recentemente pela Apple, a empresa informou que instalou um software via atualização para que os iPhones mais antigos ficassem lentos conforme a bateria ia envelhecendo. Segundo a empresa, essa medida visava entregar a melhor experiência para o usuário e aumentar a vida útil do aparelho. No

entanto, com a lentidão, o usuário acaba optando pela troca por um modelo mais novo (UOL, 2017).

Essas “portas dos fundos” permitem ainda que as empresas proprietárias possam vigiar o que o usuário faz no computador. Podem ainda copiar informações sobre movimentações bancárias, compras, sites visitados, textos escritos, imagens visualizadas etc. Esses dados podem ser usados pelas empresas ou vendidos a terceiros. Isso não significa que elas estejam fazendo, mas que, se quiserem, podem fazer. Stallman (2013) afirma que não somente as empresas se utilizam dessas funcionalidades, como também elas as repassam para governos para que possam espionar seus cidadãos.

Essa grande coleta de dados pessoais que é feita não somente em computadores pessoais, mas também em sites na internet é responsável pelo chamado Big Data[20], que é utilizado para extração de dados sobre usuários de diversas plataformas. Esses dados são tratados e vendidos a empresas e governos para que as informações em massa possam ser extraídas tanto para campanhas publicitárias quanto para manipulação política. Recentemente, o Facebook admitiu que uma grande quantidade de informações de seus usuários foi roubada e foi utilizada por candidatos à presidência da República de alguns países para manipular o resultado de eleições. Entre essas eleições, está a para presidente dos Estados Unidos (G1, 2018).

Em outro caso, o governo brasileiro, por meio do Ministério Público Federal (MPF), moveu uma ação contra a Microsoft por causa de uma funcionalidade que coleta os dados pessoais dos usuários sem a permissão expressa deles. Segundo o MPF, existe uma funcionalidade conhecida como telemetria básica que faz essas coletas. Ainda segundo o MPF, essa funcionalidade não pode ser desativada. Porém, a Microsoft alega que os dados coletados são somente para melhoria do sistema e são de controle dos usuários. Mas, nesse caso, o MPF não aceitou os argumentos e deu prazo de 15 dias para a Microsoft regularizar a situação e ainda propôs uma multa de 10 milhões de reais a título de indenização por danos morais e mais 100 mil reais por dia em caso de descumprimento (UOL, 2018). Isso mostra

[20] “Big Data é o termo utilizado para descrever grandes volumes de dados e que ganha cada vez mais relevância à medida que a sociedade se depara com um aumento sem precedentes no número de informações geradas a cada dia” (CASTRO, 2018, p. 1).

que as grandes empresas insistem na prática de coleta ilegal de informações sobre seus usuários, inclusive de órgãos públicos e governos.

Stallman (2013) afirma que os proprietários de software tentam impor cada vez mais seu controle, chegando ao ponto de proibir críticas ao desenvolvedor. Segundo ele, existe “[…] um programa proprietário para gestão de sites web que proíbe seu uso para publicar qualquer coisa que critique o desenvolvedor do programa. E neste caso perder a liberdade com o software proprietário é perder até a liberdade de expressão.”

Essas não são características de todo software proprietário, porém não é possível saber quais são os que possuem e quais os que não possuem essas características, pois o acesso ao seu código-fonte não é permitido, ficando assim o usuário sem saber em qual software ele pode de fato confiar. No entanto, Stallman (2013) alega que o que é possível afirmar é que os desenvolvedores de software são humanos e que por isso cometem erros. Sendo assim, tanto software livre quanto software proprietário possuem erros. Contudo, o software livre possibilita que a comunidade possa identificar esses erros, enquanto o software proprietário não.

Portanto, conforme afirma Stallman (2013), todo software deveria ser livre e o seu controle não deveria ser das empresas nem dos programadores, mas sim dos usuários, para que possam decidir o que eles querem executar em seus computadores, o que deveria ser de propósito geral. Ele afirma ainda que “[…] a raiz desse problema, tanto em PCs em geral como em computadores restritos, é software controlado por seu desenvolvedor. O desenvolvedor (tipicamente uma corporação) controla o que o programa faz e impede que todos os mudem” (STALLMAN, 2010, p. 228). Sendo assim, “[…] se o desenvolvedor decide colocar recursos maliciosos, mesmo um programador mestre não pode removê-los facilmente” (STALLMAN, 2010, p. 228).

Para tratar esse problema, é necessário migrar esse controle sobre o desenvolvimento do software, passando do desenvolvedor para a comunidade e para os usuários. Só assim será possível, de fato, saber o que o software realmente faz. Caso o usuário não seja um programador, ele ainda poderá usufruir dessa liberdade através da comunidade, que poderá realizar melhorias e atualizações de segurança e disponibilizá-las. O usuário pode ainda contratar um programador para realizar as modificações desejadas.

Esse poder de esconder do usuário o que o software faz, com intenções comerciais ou maliciosas, não existe no software livre:

> Com o software livre, ninguém tem o poder de criar um recurso malicioso. Como o código-fonte está disponível para os usuários, milhões de programadores estão em posição de detectar e remover o recurso malicioso e liberar uma versão melhorada. Certamente alguém vai fazer isso. Outros podem então comparar as duas versões para verificar de forma independente qual versão trata os usuários corretamente. Como um fato prático, o software livre geralmente é livre de *malware* projetado (STALLMAN, 2010, p. 228).

Então, conclui-se que, segundo o autor, o controle de desenvolvimento de software pelas empresas coloca o usuário em uma posição de vulnerabilidade e faz com que ele não consiga saber o que acontece em seu computador. O software livre pode auxiliar o usuário a controlar seu computador, dando a ele a possibilidade de verificar seu funcionamento e de aprimorar as funcionalidades.

4.4 Argumento desmoralizador

Táticas utilizadas por algumas empresas de software proprietário visam desmoralizar o movimento de software livre. Elas buscam adjetivá-lo com frases e imagens que o associe a práticas questionáveis, tais como pirataria e roubo de dados de usuários. Essas empresas criaram diversos meios para diminuir a importância desse movimento e assim "[…] faz parte também sustentar essa imagem de que software livre é inferior, mais atrasado – 'genérico' no sentido de imitação, de quebra galho" (ARAGÃO, 2015, p. 167). Juntamente com esses argumentos, essas empresas, com apoio da mídia, retratam os defensores de software livre como pessoas que não respeitam direitos autorais e que defendem a pirataria (STALLMAN, 2013).

Para Sola (2002, p. 44), o direito autoral é uma "[…] proteção dos interesses do autor e seus sucessores, em relação às suas obras criadas, sejam elas ou não literária, artística e científica". O referido autor também menciona Zabale e Beltramone, para quem é direito do autor ter o controle sobre a autorização ou a proibição de uso de sua obra e de receber uma retribuição pelo seu trabalho. Contudo, ele lembra que esses direitos são de natureza temporária e devem conciliar o interesse privado com uma utilidade social. O autor complementa

afirmando que “[…] a temporalidade do direito autoral deve atender a anseios de interesse público, motivos pelos quais sua exclusividade limita-se a um período prefixado de tempo de duração” (SOLA, 2002, p. 46). Já para Stallman (2010), os direitos autorais devem ser temporários e seu propósito é promover o progresso e não recompensar os autores. Para ele, esses direitos só são justificados quando são para o bem do público.

A violação de leis de propriedade intelectual por parte de usuários de softwares através de cópias ilegais é considerada pirataria. Na definição de Lessa (2003, p. 10), “a pirataria consiste, em primeira análise, na violação das leis de propriedade intelectual”. Sendo assim, quem compartilha ou usa software pirata está cometendo um ato ilícito por violar uma lei em vigor. Ainda segundo a autora, esse é um crime punido com detenção de até quatro anos.

O termo pirataria também é uma forma de desmoralização, pois compara alguém que utiliza software compartilhado com um pirata que invade navios para realizar assaltos. Para Stallman, esses termos são utilizados como forma de controle e de impor à sociedade as normas estabelecidas por essas empresas. Ele afirma que:

> os proprietários apresentam vários tipos de argumentos para lhes dar o poder de controlar como usamos informações: os proprietários usam palavras de difamação como “pirataria” e “roubo”, bem como terminologia especialista como “propriedade intelectual” e “dano”, para sugerir uma determinada linha de pensar ao público – uma analogia simplista entre programas e objetos (STALLMAN, 2010, p. 38).

Ainda segundo Stallman, a ideia que a sociedade faz de propriedade está relacionada a objetos materiais e se é correto alguém se apropriar de objetos de outra pessoa. Para Stallman, “[…] a informação é diferente de objetos materiais como carros e pães porque as pessoas podem copiá-la e compartilhá-la por conta própria e, se ninguém tentar detê-lo, ele pode mudá-la e torná-la melhor para si” (BETZ; EDWARDS, 1986). Nesse sentido, a cópia de determinado software não estaria incluída nesse conceito de apropriação. Porém, segundo Stallman (2010), é exatamente esse conceito que os desenvolvedores de software proprietário usam quando dizem que não se pode compartilhar um software. Em seus argumentos, eles alegam que a cópia não autorizada de um software causa prejuízos ou perdas econômicas para as empresas de software proprietário. Contudo, Stallman afirma que essa perda estaria ligada somente aos usuários que já tinham a intenção de

comprar o software e que, apesar disso, o cálculo das perdas inclui todas as cópias, inclusive as de pessoas que jamais comprariam o software (STALLMAN, 2010).

Com as leis atuais, os proprietários podem restringir o uso de software através de ameaças de penalidades e com discursos de moralidade. Stallman afirma ainda que:

> Esta linha de persuasão não é projetada para enfrentar o pensamento crítico; Está destinado a reforçar um caminho mental habitual. É elementar que as leis não decidem o certo e o errado. Todo americano deveria saber que, na década de 1950, era contra a lei em muitos estados uma pessoa negra se sentar na frente de um ônibus. Mas apenas racistas diriam que ele estava sentado no lugar errado (STALLMAN, 2010, p. 38).

Nesse sentido, Stallman (2010) acredita que a legislação protege os interesses de grupos específicos em detrimento dos interesses do restante das pessoas. São leis voltadas para garantir a manutenção do poder econômico desses grupos e que têm como pano de fundo um discurso de moralidade e de direitos naturais. Stallman (2010, p. 38) afirma que os chamados direitos naturais fazem com que muitas vezes os autores de software reivindiquem "[…] uma conexão especial com os programas que escreveram e que, como resultado, seus desejos e interesses em relação ao programa simplesmente superam aqueles de qualquer outra pessoa – ou mesmo aqueles de todo o resto do mundo".

4.5 Programação, geração de renda e motivação

Desenvolver software é uma atividade que pode contribuir para a sociedade. Contudo, nem sempre esse desenvolvimento realmente realiza uma contribuição relevante. Stallman (2013) afirma que "[…] desenvolver um software livre é uma contribuição para a sociedade, porém desenvolver software privado não é uma contribuição e sim um golpe com a intenção de se apoderar das pessoas". O desenvolvimento de software livre é um caminho para uma sociedade mais colaborativa, em que os indivíduos tenham plena consciência do seu papel como parte de um todo.

Os programadores que desenvolvem software livre podem contribuir para a libertação da sociedade de um sistema que valoriza as atividades não produtivas e que busca alcançar cada vez mais lucro. Portanto, desenvolver software livre é um trabalho social importante.

Segundo Stallman (2010), o desenvolvimento de softwares livres pode levar, de forma indireta, a uma sociedade na qual o esforço no trabalho poderá ser reduzido, permitindo maior aproveitamento do tempo pelo trabalhador. Segundo o autor:

> nós já reduzimos consideravelmente a quantidade de trabalho que toda a sociedade deve fazer pela sua produtividade real, mas apenas um pouco disso traduziu em lazer para os trabalhadores porque é necessária muita atividade não produtiva para acompanhar a atividade produtiva. As principais causas disso são a burocracia e lutas isométricas contra a concorrência. O software livre reduzirá muito estes drenos na área de produção de software. Devemos fazer isso, para fins técnicos, ganhos de produtividade para se traduzir em menos trabalho para nós (STALLMAN, 2010, p. 36).

Segundo Stallman (2010), um importante argumento contra o software livre é que, se o software não for cobrado, os programadores perderão seus empregos e sua fonte de renda. No entanto, esse argumento pode ser refutado, pois software livre não significa software gratuito. Assim como afirmado no capítulo 3, o software livre "[…] vem com permissão para que qualquer pessoa utilize, copie e/ou distribua, seja por escrito ou com modificações, gratuitamente ou por uma taxa […]". Portanto, nada impede que quem o desenvolveu ou qualquer outra pessoa possa vendê-lo. Por exemplo, um programador pode utilizar um software livre para iniciar um negócio em que ele customiza, implanta e dá suporte a esse software, cobrando pelo seu trabalho. Em outro exemplo, alguém pode desenvolver um programa como software livre, distribuí-lo gratuitamente, solicitando doações e cobrar por seu suporte, ou até mesmo vender a mídia física para quem queira recebê-lo pelos correios. Além disso,

> […] existem muitas maneiras pelas quais os programadores poderiam ganhar a vida sem vender o direito de usar um programa. Este caminho é habitual agora porque traz mais dinheiro para os programadores e empresários, não porque seja a única maneira de ganhar a vida. É fácil encontrar outras maneiras se você quiser encontrá-las. Aqui estão alguns exemplos. A venda de serviços de ensino, mão de obra e manutenção também poderiam empregar programadores. Pessoas com novas ideias poderiam distribuir programas como *freeware*, pedindo doações de usuários satisfeitos ou venda de serviços de mão de obra (STALLMAN, 2010, p. 35).

Para o autor, não há nada de errado em ganhar dinheiro com desenvolvimento de software, desde que não se usem meios destrutivos para isso. E, segundo ele, ganhar dinheiro com a restrição do uso de software é destrutivo, pois reduz a quantidade de riqueza que a comunidade poderá extrair desse

software. A utilização deliberada de meios destrutivos causa destruição deliberada. Se todos os cidadãos usarem meios destrutivos para ganhar dinheiro, todos se tornarão pobres, pois todos destruirão e serão destruídos ao mesmo tempo (STALLMAN, 2010).

Diante do raciocínio de alguns de que software livre prega que não se deve recompensar a criatividade de quem desenvolve software, Stallman alega que:

> Se alguma coisa merece uma recompensa, é contribuição social. A criatividade pode ser uma contribuição social, mas apenas na medida em que a sociedade é livre para usar os resultados. E se os programadores merecem ser recompensados pela criação de programas inovadores, do mesmo modo, eles merecem ser punidos se restringirem o uso desses programas (STALLMAN R, 2010, p. 32).

É natural que quem desenvolve software se sinta desejoso de ser reconhecido e recompensado pelo seu esforço e pelo seu trabalho, porém este "[…] desejo de ser recompensado pela criatividade de alguém não justifica privar o mundo em geral de tudo ou parte dessa criatividade" (STALLMAN, 2010, p. 32). Ele afirma ainda que a recompensa deve ser dada para quem produz coisas que beneficiam a sociedade e não para quem quer apenas enriquecer. Em um trecho de uma entrevista para o site GNU.org, Stallman disse o seguinte:

> Eu gostaria de ver as pessoas sendo recompensadas por escrever software livre e por encorajar outras pessoas a usá-lo. Eu não quero ver pessoas recebendo recompensas por escrever softwares proprietários porque isso não é realmente uma contribuição para a sociedade (BETZ; EDWARDS, 1986).

A programação é uma atividade divertida para muitas pessoas, elas a fazem por puro prazer em resolver problemas e desafios e, quando conseguem, querem compartilhar suas conquistas com outras pessoas, por orgulho e reconhecimento, e outras vezes por dinheiro. Isaacson (2014, p. 379) sustenta que "[…] os *hackers* também são motivados, em grande parte, pelo respeito que podem ganhar aos olhos de seus pares ao dar contribuições sólidas […]". Nesse sentido, Linus Torvalds (2001) alega que o dinheiro é um motivador muito poderoso quando você está em um nível de subsistência, porém, quando se está em um nível de entretenimento, ele não é tudo. E a programação em si já é um fator motivador para muitos programadores. "A programação tem um fascínio irresistível para algumas pessoas, geralmente as pessoas quem são melhores nisso" (STALLMAN, 2010, p. 34).

Quando os problemas são difíceis, os programadores gostam de mostrar como foram resolvidos e por isso disponibilizam o código para apreciação e melhoria.

O fato de desenvolver código que pertence a uma corporação e que não pode ser compartilhado faz com que essas pessoas se sintam “[…] infelizes com a comercialização de sistemas e programas. Pode permitir-lhes ganhar mais dinheiro, mas exige que eles se sintam em conflito com outros programadores em geral em vez de se sentir como camaradas” (STALLMAN, 2010, p. 29).

Para a comunidade *hacker*, é muito importante o compartilhamento de códigos e programas ou a reciclagem de código. Para eles, “[…] o ato fundamental de amizade entre programadores é o compartilhamento de programas” (STALLMAN, 2010, p. 29).

4.6 Liberdades essenciais

O principal objetivo do software livre é permitir que os usuários possam tomar as decisões relacionadas à sua utilização, tornando possível para eles a possibilidade de executar, copiar, distribuir e estudá-lo, fazendo melhorias quando necessário. Conforme dito no capítulo 3, o software livre respeita as liberdades individuais dos usuários. Sendo assim, para que esses usuários possam ter a possibilidade de desfrutar de suas liberdades individuais durante o uso de determinado software, a Free Software Foundation definiu algumas regras que identificam o software como livre, nomeando-as como liberdades essenciais. Sendo elas as seguintes:

> • A liberdade de executar o programa, para qualquer propósito (liberdade 0).
> • A liberdade de estudar como funciona o programa e mudá-lo para fazê-lo executar o que você deseja (liberdade 1). O acesso ao código-fonte é uma condição prévia para esta liberdade.
> • A liberdade de redistribuir cópias para que você possa ajudar o seu próximo (liberdade 2).
> • A liberdade de distribuir cópias de suas versões modificadas para outros (liberdade 3). Ao fazer isso, você pode dar a toda a comunidade uma chance de se beneficiar de suas mudanças. O acesso ao código-fonte é uma condição prévia para esta liberdade. (STALLMAN, 2010, p. 3)

4.7 Dano causado pelo software proprietário

Conforme estudado no capítulo 3, o software proprietário limita sua utilização por parte do usuário através de restrições na sua forma de distribuição e de desenvolvimento. Stallman acredita que essa limitação pode causar danos sociais consideráveis, pois, segundo ele, impedem o desenvolvimento humano e a cooperação. O autor afirma ainda que existem três níveis de danos sociais que advêm das restrições impostas pelas grandes corporações e pelos governos através do software proprietário e das patentes de software. Em um primeiro momento, menos pessoas utilizariam o software, e as que usam não podem adaptá-lo ou corrigir possíveis erros; e, por fim, ninguém poderá aprender com o software ou basear novos trabalhos nele. Stallman acredita que cada uma dessas restrições tem uma forma concomitante de dano psicossocial, que, segundo ele:

> […] se refere ao efeito que as decisões das pessoas têm sobre seus sentimentos subsequentes, atitudes e predisposições. Essas mudanças nos modos de pensar das pessoas então terão um efeito adicional sobre suas relações com seus concidadãos e pode ter consequências materiais (STALLMAN, 2010, p. 46).

A indústria de software proprietário obstrui o desenvolvimento de programas, pois produz apenas aqueles que são vendáveis, comerciais e que têm seu uso restrito a usuários que pagaram pela licença. O autor assegura que essas limitações não trazem benefícios para a sociedade nem para os desenvolvedores. Porém, prejudicam os usuários. Ele apresenta como exemplo um usuário que precisa de determinado software que está disponível para venda. Contudo, nesse caso, esse usuário não pode pagar pela licença e por causa disso decide renunciar ao seu uso. Para Stallman (2010, p. 46), “[…] quando um usuário opta por pagar, esta é uma transferência de riqueza de montante zero entre duas partes. Mas cada vez que alguém escolhe renunciar ao uso do programa, isso prejudica essa pessoa sem beneficiar ninguém.” Portanto, proibir o uso de software através de cobrança de uma taxa não beneficia o desenvolvedor, mas sim prejudica o usuário que deixa de usá-lo.

Além disso, o software proprietário não estimula a coesão social, pois impede que os cidadãos possam compartilhar conhecimento aplicado em tecnologia da informação. Para Stallman, “[…] tentar dominar o conhecimento, tentar controlar se

as pessoas podem usá-lo ou tentar impedir outras pessoas de compartilhá-lo é sabotagem" (BETZ; EDWARDS, 1986).

Stallman afirma que a indústria de software põe o usuário em uma situação em que ele tem de escolher entre dois males. Um deles é deixar de compartilhar com sua comunidade o conhecimento aplicado em ciência da computação contido em um software proprietário. O outro mal consiste em romper com um contrato assinado e compartilhar esse software com a comunidade, restringindo assim a disseminação do conhecimento utilizado para o desenvolvimento do software. Ainda segundo ele, mesmo que esses contratos sejam um mal para a sociedade, rompê-los não chega a ser bom, pois faz com que o usuário fique na posição de transgreção da lei de direitos autorais.

Sendo assim, o software proprietário causa um "[…] dano psicossocial associado ao prejuízo material do uso desencorajador do programa. Os programadores também sofrem dano psicossocial sabendo que muitos usuários não irão ter permissão para usar os softwares desenvolvidos por ele" (STALLMAN, 2010, p. 47). O autor acredita que se algum mal tiver que ser feito que seja feito contra as empresas de software, pois elas estão tentando controlar o que o usuário pode fazer. No entanto, existe uma forma de escapar desse dilema: abandonar o uso do software proprietário e adotar o uso do software livre (STALLMAN, 2010). A utilização de software livre impede que os danos sociais ocorram, por causa das liberdades essencias, que são a base da filosofia do software livre, e porque todo software livre é desenvolvido sob a licença GPL.

Os desenvolvedores de software podem contribuir para evitar os danos sociais causados pelo software proprietário. Nesse sentido, eles podem trabalhar com software livre, criando softwares úteis e dando suporte a usuários com menor conhecimento. Stallman afirma que, se o desenvolvedor não está disposto a criar software livre, "[…] então é melhor não fazer nada do que desenvolver um software proprietário, porque se não faz nada não causa dano" (STALLMAN, 2013).

4.8 Superar a inércia social

Para Stallman (2010, p. 42), "o principal obstáculo ao triunfo da liberdade do software é a inércia social". Para o autor, existem diversas formas de inércia social, entre elas ele cita o fato de os usuários preferirem a comodidade de um software

pronto ao trabalho de desenvolver um software livre. De acordo com Kantowitz, Roediger III e Elmes (2006, p. 3), a inércia social está relacionada com o fato de que “muitas pessoas em grupos parecem dispostas a deixar que poucos façam o trabalho”. Nesse sentido, o usuário deixa de exercer seu direito de escolha por comodidade.

Segundo Stallman (2010), a inércia social consiste em pessoas que cederam a ela e que se tornaram parte da pressão social exercida sobre outras pessoas. Ele afirma que o que causa essa inércia social é a estratégia das empresas de software, que restringem o uso de computadores a sistemas e softwares proprietários, fazendo com que as pessoas se vejam obrigadas a utilizarem esses softwares e, depois de um tempo, elas se habituam e não conseguem mais mudar. Essas empresas trabalham contra a disseminação do livre conhecimento, desencorajando a produção de ferramentas úteis e que ao mesmo tempo sejam livres de amarras e possam servir à sociedade. Stallman afirma que:

> […] o princípio do capitalismo é a ideia de que as pessoas conseguem ganhar dinheiro produzindo coisas e, portanto, são encorajadas a fazer o que é útil, automaticamente, por assim dizer. Mas isso não funciona quando se trata de possuir conhecimento. Eles são encorajados a não fazer realmente o que é útil, e o que realmente é útil não é encorajado (BETZ; EDWARDS, 1986).

Para Stallman (2010), outra estratégia das empresas de software é a de viciar estudantes nas escolas e universidades, por meio de incentivo financeiro para as escolas adquirirem seus softwares e obrigarem seus alunos a utilizá-los, tornando-os assim dependentes desses softwares. Por exemplo, a Microsoft possui um programa educacional chamado Office 365 Education, que permite a alunos e professores de instituições acadêmicas utilizarem alguns produtos via web ou até mesmo instalarem em computadores, sem custo, enquanto eles estiverem na instituição (MICROSOFT, 2018). Após o vínculo institucional, eles devem pagar pelos produtos se quiserem continuar usando. Para Stallman, as “[…] escolas que ensinam Windows, produzem graduados que estão acostumados a usar o Windows, e isso encoraja as empresas a usar o Windows” (STALLMAN, 2010, p. 245).

Na visão de Stallman (2010), a principal forma de lutar para acabar com a inercia social consiste na escola recusar-se a ensinar software proprietário a seus alunos. Para ele, isso é possível, pois essas não vão precisar dos descontos

concedidos pelas empresas para adquirirem softwares educacionais[21], pois software livre pode ser conseguido de graça. Outra forma consiste em cada usuário resistir ao uso de software proprietário, superando as dificuldades iniciais de se aprender a utilizar um novo recurso computacional. Isso pode ser conseguido através de treinamentos e persistência. Existem muitos treinamentos gratuitos para que eles possam descobrir que software livre pode fazer o mesmo ou até mais que o proprietário.

4.9 Aproximação para o controle

Algumas das grandes empresas de software já perceberam o potencial do software livre e estão cada vez mais se aproximando desse movimento, porém o que se questiona é a intenção delas por trás dessa aproximação.

Empresas como IBM, Microsoft e Amazon, que sempre estiveram na contramão desse movimento, se aproximaram dele e apresentaram suas soluções para softwares “livres”.

A Microsoft incentiva a utilização de software livre em sua plataforma em “nuvem” para que os desenvolvedores e as empresas migrem seus sistemas livres para essa plataforma. Com o discurso de melhor infraestrutura, ele consegue controlar novamente os usuários, mesmo que esses estejam usando software livre, e com isso buscam novamente a obtenção de lucro. Isso se dá nessa plataforma, pois o usuário ou a empresa não tem o controle total de sua informática, tendo em vista que os mecanismos de virtualização não são transparentes e todos os dados do usuário terão necessariamente de transitar pelos servidores da Microsoft. Outro fator importante é que, se o usuário deixa de pagar, ele perde acesso aos serviços e com isso seus sistemas deixam de funcionar (STALLMAN, 2013).

Outras iniciativas da Microsoft para aproximar-se do software livre vêm sendo divulgadas. Por exemplo, a empresa se filiou à Linux Foundation e se tornou um membro Platinum, que é o maior nível de parceria dessa organização. Segundo Jim Zemlin (MICROSOFT, 2016), “a Microsoft se tornou o principal colaborador em muitos projetos e vemos que a empresa intensificou seu envolvimento e

[21] Softwares educacionais são “os programas utilizados em processos administrativos escolares ou em contextos pedagógicos, sendo eles categorizados como: software educativo e software aplicativo” (MORAIS, 2003, p. 21).

comprometimento no desenvolvimento aberto". Outro anúncio recente da Microsoft é a criação de sua própria distribuição Linux, o Azure Sphere OS, com a finalidade de ser utilizado na internet das coisas[22].

Da mesma forma, a Amazon possui uma plataforma de virtualização em que as empresas e os usuários podem pagar para manter seus sistemas em funcionamento, seja em software proprietário ou software livre. O problema é que não existe transparência para esse usuário sobre como seus dados são armazenados e se esses dados estão acessíveis para a Amazon. Existem contratos que supostamente protegeriam o usuário, porém este não possui meios para auditar essa proteção.

A utilização de software como código-fonte aberto, desenvolvido por empresas que historicamente são adversárias do movimento software livre, pode levar o usuário ao erro de confundir as duas modalidades de software e torná-lo novamente um usuário dependente desses softwares e dessas empresas.

4.10 Progresso da computação e a inclusão digital

A computação se desenvolveu muito rapidamente ao longo da segunda metade do século XX. A humanidade saiu de um computador que ocupava um andar inteiro e chegou a computadores que cabem na palma da mão e são milhares de vezes mais performáticos que o primeiro. O computador deixou de estar confinado a grandes universidades e a empresas e passou ao bolso dos cidadãos. Esse rápido avanço trouxe consequências boas e ruins, trouxe imensas vantagens para a humanidade, no sentido de melhoria de qualidade de vida e de melhoria nas condições de trabalho. Porém, muitas vezes, somos levados a questionar até que ponto o progresso da computação é bom ou ruim.

Um progresso que ajude a humanidade a aliviar o trabalho pesado, a salvar vidas e explorar o espaço, a avançar em outras áreas da ciência, com certeza não é questionável, porém, se esse progresso avança sobre as mentes, controlando e manipulando pensamentos, limitando o intelecto e os próprios avanços, talvez ele deva ser questionado.

[22] A internet das coisas, em poucas palavras, nada mais é que uma extensão da internet atual, que proporciona aos objetos do dia a dia (quaisquer que sejam), mas com capacidade computacional e de comunicação, se conectarem à internet (SANTOS; SILVA; CELES, 2017).

Mesmo com todo o progresso na tecnologia da informação, diversas pessoas no mundo ainda são excluídas digitalmente, pois não têm acesso a computadores nem à internet. Os países menos desenvolvidos são os que apresentam a maior taxa de exclusão digital. Para Souza (2008, p. 23), “a exclusão digital pode significar um aprofundamento ainda maior da divisão entre as populações dos países ricos e dos países pobres dificultando o processo de desenvolvimento do Terceiro Mundo”.

Da mesma forma, a inclusão digital pode ser um instrumento de libertação do indivíduo e uma forma de educar a sociedade, porém não é qualquer inclusão digital que é capaz de libertar. Silva *et al*. (2005, p. 30) afirmam que:

> Tem-se, então, como fundamental, que a inclusão digital deve ser vista sob o ponto de vista ético, sendo considerada como uma ação que promoverá a conquista da “cidadania digital” e contribuirá para uma sociedade mais igualitária, com a expectativa da inclusão social.

Nesse sentido, inclusão digital que se dê através de softwares que fazem de seus usuários reféns de seu conteúdo e de suas licenças não liberta, mas sim, ao contrário, torna-os escravos de certas ideias e conceitos. Além disso, fazem com que esses usuários se tornem dependentes desses programas. Richard Stallman (2013) disse em uma entrevista ao jornal *Zero Hora*:

> Eu não acho que inclusão digital seja necessariamente boa. Depende no tipo de sociedade digital em que vivemos. Se é uma sociedade livre, que respeita nossa liberdade, então é boa. Se é injusta e tirânica, como a que Agência Nacional de Segurança dos Estados Unidos proporciona, o povo está melhor sem inclusão digital. O que deveríamos fazer é lutar por uma sociedade digital livre (IHU, 2013).

Software livre pode proporcionar uma inclusão digital que realmente proporcione aos usuários liberdade de escolha e agregue conhecimento. Esse tipo de inclusão digital é emancipadora e faz com que as pessoas evoluam no sentido de estar incluídas no ambiente digital.

5 CONTRIBUIÇÕES DO SOFTWARE LIVRE PARA EDUCAÇÃO: EDUCAÇÃO CÍVICA E EMANCIPAÇÃO DO INDIVÍDUO

Neste capítulo serão indicados aspectos relacionados à utilização de tecnologias em ambiente educacional. A discussão se concentrará no uso de software livre como ferramenta de ensino e instrumento emancipador. Nesse sentido, será analisado como o software livre pode, em princípio, contribuir para educação e para emancipação do indivíduo. Para que essa análise possa ser realizada, serão explicitadas as definições de conhecimento, de educação e educação profissional e de educação tecnológica. Por fim, será abordado o uso de tecnologias e de software livre na educação.

5.1 Reflexões a respeito de educação

A partir da definição de conhecimento já apresentada, é possível refletir sobre formas utilizadas pela humanidade para tentar transmitir os conhecimentos adquiridos ao longo dos anos. Para isso, faz-se necessária a análise de algumas definições de educação. Para os gregos antigos, o conceito de educação era a Paideia, a formação do homem, que passava pela educação do corpo e da mente. Em seu livro *Paideia – a formação do homem grego,* W. Jaeger traz a seguinte definição do conceito de Paideia:

> O conceito que originariamente designava apenas o processo da educação como tal estendeu ao aspecto objetivo e de conteúdo a esfera do seu significado, exatamente como a palavra alemã *Bildung* (formação) ou a equivalente latina cultura, do processo da formação passaram a designar o ser formado e o próprio conteúdo da cultura, e por fim abarcaram, na totalidade, o mundo da cultura espiritual: o mundo em que nasce o homem individual, pelo simples fato de pertencer ao seu povo ou a um círculo social determinado. A construção histórica deste mundo atinge o seu apogeu no momento em que se chega à ideia consciente da educação. Torna-se assim claro e natural o fato de os gregos, a partir do século IV, quando este conceito encontrou a sua cristalização definitiva, terem dado o nome de Paideia a todas as formas e criações espirituais e ao tesouro completo da sua tradição […] (JAEGER, 1986, p. 245, 246).

Com base na etimologia da palavra “educação”, ela se origina de termos latinos, como os verbos *educare* e *educere*. *Educere* significa conduzir para fora. Nessa forma de educação, percebe-se a figura do mestre que procura extrair do pupilo o conhecimento. Já *educare* tem como significado amamentar; criar,

alimentar, e tem proximidade com a palavra *cuore* (coração) (HELPA, 2015). Nesse sentido, educar está ligado à condução ao conhecimento e, ao mesmo tempo, à extração do conhecimento interior.

Sem pretender esgotar os conceitos sobre educação, a ideia de Paulo Freire a esse respeito é: “[…] sempre uma certa teoria do conhecimento posta em prática […]” (FREIRE, 1982). O educador brasileiro afirma que não é possível a prática da educação sem que se tenha algo a aprender, pois sempre haverá a relação entre sujeito e objeto.

Refletindo sobre as definições de educação, conclui-se que a educação moderna está mais ligada à perpetuação de linhas de pensamento e à inculcação de ideologias das classes dominantes do que na formação do homem (SANTOS, 1980). As escolas, como aparelhos ideológicos do Estado, efetivam esse papel. Conforme afirma Santos (1980, p. 24): “Desta forma, a escola, como responsável pela transmissão da educação formal, passa a ser agente de reprodução de uma ideologia burguesa.” Sendo assim, entende-se que a educação atual não prioriza a liberdade de pensamento, tendo em vista que, quando se transmite certos conhecimentos, também são transmitidas ideias que nada tem a ver com o conhecimento que se quer transmitir, mas sim com ideologias. Para Carvalho (2015, p. 36), “a educação não é neutra e, por isso, requer que o docente trabalhe a relação do homem com a sua realidade de maneira planejada e organizada […]”. Portanto, os educadores devem fazer escolhas que garantam uma educação em que o indivíduo tenha liberdade, não somente no sentido de direitos, mas liberdade de pensamento. Nesse sentido, o papel dos educadores vai além do conteúdo apresentado em sala de aula, pois ele deve “[…] contribuir significativamente na formação de um aluno que saiba refletir, tomar decisões, ser crítico e autônomo” (CARVALHO, 2015, p. 36).

Paulo Freire apresenta, em seu texto de 1967, o tipo de educação que, segundo ele, possibilita ao indivíduo atingir a libertação. Para ele, esta educação deve ser a:

> Educação que, desvestida da roupagem alienada e alienante, seja uma força de mudança e de libertação. A opção, por isso, teria de ser também, entre uma “educação” para a “domesticação”, para a alienação, e uma educação para a liberdade. “Educação” para o homem-objeto ou educação para o homem-sujeito (FREIRE, 1967, p. 36).

A educação, que por vezes pode ser libertadora, pode também levar o indivíduo à alienação. A alternativa é fazer uma opção por uma educação que leve o ser humano à reflexão e o conduza rumo à liberdade de pensamento.

5.2 Educação profissional e tecnológica

As políticas de governo aplicadas à educação se refletem na educação profissional e tecnológica, pois essa sofre constantes alterações em seus objetivos fundamentais. A cada novo governo pacotes de alterações são apresentados como melhorias e avanços, porém, muitas vezes, essas alterações são apenas para atender determina vertente política ou ideológica.

A educação profissional e tecnológica objetiva a formação de trabalhadores, de pessoas preparadas para o trabalho, visando ensinar e aprimorar as competências técnicas. Ao se verificar o texto sobre o tema no portal do MEC, tem-se a ideia dos reais objetivos da educação profissional e tecnológica:

> A educação profissional de nível tecnológico, integrada às diferentes formas de educação, ao trabalho, à ciência e à tecnologia, objetiva garantir aos cidadãos o direito à aquisição de competências profissionais que os tornem aptos para a inserção em setores profissionais nos quais haja utilização de tecnologias. (MEC, 2002, p. 465).

Ao observar o passado, constata-se que, desde o princípio, a educação no Brasil se preocupou mais em formar trabalhadores. Nas primeiras escolas de aprendizes artífices criadas no início do século XX, o objetivo era a formação de mão de obra e a educação moral e cívica dos indivíduos desfavorecidos da fortuna. Isso se evidencia neste trecho do Decreto nº 7.566, de 23 de setembro de 1909, da criação das escolas:

> Que o argumento constante da população das cidades exige que se facilite ás classes proletárias os meios de vencer as dificuldades sempre crescentes da luta pela existência;
>
> Que para isso se torna necessário, não só habilitar os filhos dos desfavorecidos da fortuna com o indispensável preparo técnico e intelectual, como fazê-los adquirir hábitos de trabalho profícuo, que os afastará da ociosidade ignorante, escola do vicio e do crime;
>
> Que é um dos primeiros deveres do Governo da Republica formar cidadãos uteis á Nação […] (BRASIL, 1909).

Com a criação das escolas de aprendizes artífices, pretendeu-se que a educação tome para si o papel de formação para o trabalho – formação importante para os ideais positivistas da nova República. “É assim que podemos relacionar aquele momento específico em nossa história e suas determinações sobre o tipo de ensino profissionalizante que o Estado oficializava, com o desenvolvimento mais geral do capitalismo” (BRANDÃO, 1999, p. 21). Desde então, apesar de muitas mudanças, a educação ainda vem sendo tratada pelos políticos como um negócio vantajoso na hora dos discursos e um laboratório onde se “experimentam” os programas de governo.

> Um breve retrospecto histórico nos indica uma trajetória de interrupções dos projetos societários que postulavam as reformas estruturais e os investimentos em educação, ciência e tecnologia, condições necessárias à constituição efetiva de uma nação soberana, mediante ditaduras e golpes (FRIGOTTO, 2007, p. 1136).

O tipo de educação profissional e tecnológica que vem sendo desenvolvido ao longo dos anos ainda mantém o ideal de formação de trabalhadores, tendo em vista os currículos dos cursos. Sendo assim, pode-se questionar: qual o *telos* da educação profissional e tecnológica? Para Brandão (2014, p. 1):

> A educação tecnológica deverá implicar a reflexão que começa no ato da produção dos artefatos, na aplicação da técnica, até o ato da utilização e consumo dos artefatos ou dos produtos gerados por estes. Desnaturalizar os processos do produzir ao consumir mediados pela técnica deve ser objetivo de uma verdadeira educação tecnológica.

Contudo, além de desnaturalizar o processo produtivo, a educação deve formar indivíduos que pensem por si só e não meros detentores de técnicas criadas por outros. Jean Piaget (1984) afirma que a educação tem por principal objetivo a criação de seres humanos com capacidades próprias para fazer cosias novas, deixando, assim, de serem meros repetidores de gerações anteriores. Portanto, devem ser seres criativos, inventivos e descobridores. O autor ainda apresenta um segundo objetivo para educação: o de formar mentes críticas e que tenham a capacidade de verificar, de conferir e não somente de aceitar o que é posto para eles como verdade.

Existe hoje a necessidade de se levantar uma discutição a respeito dos rumos que a educação profissional e tecnológica deve seguir. Se ela deve se manter na formação estritamente técnica, formando trabalhadores preparados, ou se ela deve

formar homens e mulheres pensantes, capazes de criarem e de refutarem ideias. Esses seres humanos serão os que criarão as tecnologias do futuro, as quais também farão parte da educação.

5.3 Uso de tecnologias na educação

O uso de tecnologias na educação vem aumentando ao longo dos anos. De acordo com Oliveira, Moura e Sousa (2015, p. 76), “cada vez mais a tecnologia se faz presente na escola e no aprendizado do aluno, seja pelo uso de equipamentos tecnológicos seja por meio de projetos envolvendo educação e tecnologia”. Na reflexão desenvolvida por Souza (2016, p. 15), a tecnologia está cada vez mais presente na sociedade em geral e nas diversas atividades humanas e, portanto, “[…] sua presença é inevitável e a escola deve estar dentro desse contexto, apropriando-se do conhecimento do uso dos recursos tecnológicos, que possivelmente poderá ser uma importante ferramenta pedagógica”. Nesse sentido, Carvalho (2015, p. 35) afirma que ”[…] a educação, hoje, está mergulhada numa sociedade da informação, uma sociedade em rede, permeada por tecnologias em constante evolução e mudança”. Já Alencar (2007) acredita que a educação sempre se utilizou de técnicas e de tecnologias para seus propósitos. Segundo ele:

> Na perspectiva teórico-filosófica com a qual defendemos o conceito de técnica e tecnologia, podemos dizer que nunca existiu uma educação que se visse desvinculada de certa técnica e de certa tecnologia. Sempre, em toda história da didática, usamos uma “forma de fazer as coisas” ou um “conjunto de formas de fazer as coisas” para ensinar e também para prender. Usamos técnicas e tecnologias (ALENCAR, 2007, p. 35).

Contudo, nem toda tecnologia é necessariamente benéfica no que diz respeito à educação e à transmissão de conhecimento. Toda tecnologia carrega consigo uma ideologia ligada ao seu criador. É o que afirma Alencar, quando diz que “os usos das diversas tecnologias estão sempre permeados pela ideologia; não se pode negligenciar isso” (ALENCAR, 2007, p. 37). Essas questões ideológicas devem ser avaliadas pelos educadores, pois, segundo Cruz *et al*. (s.d., p. 2), “[…] o mau uso da tecnologia na educação pode ser um grande agravante no processo de desenvolvimento da criatividade e da síntese de informação”.

Por conseguinte, o aumento da utilização de tecnologias que, ao contrário de facilitar o desenvolvimento intelectual e a criatividade, submete o indivíduo a um

processo de automatização, faz com que seja necessário repensar o uso de certas tecnologias em detrimento de outras. Tecnologias estas que venham a suprir a necessidade de ampliação do conhecimento, sem que, com isso, os indivíduos se tornem alienados. Nessa lógica, tecnologias que causam dependência devem ser evitadas e, portanto, devem dar lugar a outras que possibilitem aos indivíduos emanciparem-se e tomarem suas próprias decisões.

Como alternativa, o uso de tecnologias livres na educação pode contribuir para a liberdade individual e de pensamento. Segundo Aguado (2012, p. 31):

> Para que a luta pela libertação seja justificável, ela não deve se dar apenas nas coisas corriqueiras como liberdade de ir e vir, liberdade de expressão, mas também pela liberdade de pensar e refletir, de lutar e ser agente ativo nas escolhas e decisões – protagonistas em suas histórias.

Assim, a discussão passa por descobrir quais as tecnologias propiciam aos indivíduos a capacidade de se aproximar dessas liberdades.

Atualmente, a educação necessita cada vez mais da utilização de tecnologias de informática para conseguir atrair o interesse dos alunos e possibilitar uma maior interação entre conhecimento e tecnologia. “A anexação do computador e da internet na vida dos alunos trouxe uma avalanche de informações que as escolas e os professores, muitas vezes, não estão preparados para absorver” (OLIVEIRA; MOURA; SOUSA, 2015, p. 76). Desse modo, faz-se necessária a utilização de programas de computadores cujo conteúdo seja atrativo e cujo formato acompanhe as tendências de liberdade que a internet tem hoje. Oliveira, Moura e Sousa (2015, p. 76) afirmam que: “A utilização de recursos tecnológicos no processo de ensino, é cada vez mais necessária, pois torna a aula mais atrativa, proporcionando aos alunos uma forma diferenciada de ensino.”

No passado, a educação foi influenciada por diversas mudanças no mundo, tais como invenções, guerras ou revoluções. Como foi o caso da revolução russa, que levou ao poder uma classe social totalmente oposta à anterior, saindo de uma educação elitista, voltada para a classe burguesa e para a monarquia, e criando currículos mais adequados à classe trabalhadora (LOMBARDI, 2017).

Desde o uso de projetores até o uso de lousas inteligentes, a utilização de tecnologias auxilia o professor na difícil tarefa de prender a atenção do aluno e de motivá-lo. A criação das cartilhas e dos livros didáticos foi um grande avanço

tecnológico na forma de ensinar. Esses instrumentos possibilitaram ao professor contar com ideias de outros educadores para auxiliá-lo nas aulas.

Outra importante contribuição da tecnologia foi a introdução dos mimeógrafos, pois possibilitou ao professor entregar ao aluno um conteúdo impresso feito por ele mesmo. O uso de canetas para lousa também melhorou a condição do educador, pois a utilização de giz gerava nos professores alergias que, por vezes, os incapacitavam para o trabalho. A introdução de videoaulas e documentários trouxeram novas oportunidades de conhecimentos aos alunos, entrando em contato com outras formas de aprender e com outros tipos de conhecimento. Todas essas tecnologias tiveram sua importância histórica e pedagógica e foram revolucionárias em sua época, porém, "[…] nos últimos anos presenciamos a difusão de um artefato tecnológico, uma ferramenta complexa que está se expandindo, tomando conta de praticamente todas as instâncias educacionais, o computador" (JUCA, 2006, p. 23).

O uso de computadores nas escolas contribuiu para uma melhor administração escolar com a utilização de softwares educacionais. De acordo com a definição de Oliveira (2001), um software educacional é uma ferramenta de informática que pode ser utilizada adequadamente pela escola, mesmo que esta não tenha sido produzida com a finalidade de uso no sistema escolar. Contudo, essa não foi a única contribuição que os computadores trouxeram para as escolas. Com a aquisição por parte dos governos dos laboratórios de informática, uma nova ferramenta foi adotada pelos professores, o software educativo[23].

Esses programas de computador são desenvolvidos para favorecer os processos de ensino, especialmente no que diz respeito à construção do conhecimento relativo a um conteúdo didático. Por exemplo, o ensino de matemática ou física através de jogos, com ou sem a mediação do professor. Esses jogos podem apresentar situações nas quais o aluno tenha que fazer escolhas que o levem à solução de determinados problemas, de acordo com a disciplina que se deseja ensinar. Isso faz com que o aluno absorva o conhecimento de uma forma que, para ele, seja divertida (JUCA, 2006).

Nesse sentido, o uso do software educativo pode contribuir para melhorar a compreensão por parte dos alunos de determinados conteúdos considerados difíceis

[23] O software educativo é uma das classes do software educacional, tendo ele como objetivo principal o de facilitar o processo de ensino-aprendizagem, fazendo com que o aluno construa determinado conhecimento relativo a um conteúdo didático (MORAIS, 2003, p. 22).

de serem ensinados, pois esses softwares podem ensinar de forma lúdica, seja através de jogos, seja através de histórias e desenhos animados. Esse tipo de abordagem prende a atenção do aluno, pois é um ambiente no qual ele está familiarizado.

Conforme afirmado no capítulo 4, na seção “Progresso da computação e a inclusão digital”, a inclusão digital pode ser um instrumento de libertação do indivíduo, e por isso é também papel da escola promover essa inclusão. Isso porque o uso de software na escola vai proporcionar para muitos dos alunos o seu primeiro contato com a informática. Esse contato é importante, pois possibilita uma expansão das oportunidades para que este adquira novos conhecimentos e para ampliar seu desenvolvimento intelectual.

O uso da internet nos processos educacionais é também um importante instrumento de inclusão, pois possibilita tanto ao aluno quanto ao professor a exploração de conhecimentos difundidos por várias partes do mundo. Além disso, “[…] as TIC[24] materializam no contexto social e escolar uma diversidade de suportes comunicativos e linguagens que requerem de professores e alunos a aquisição de novas competências e habilidades para continuar conectadas com o mundo” (FERNANDES, 2011, p. 3). Contudo, esse conhecimento precisa ser filtrado e direcionado para os fins pedagógicos pretendidos, para que essas competências e habilidades sejam utilizadas de forma correta.

5.4 Software livre e educação: uma relação recíproca

O uso de tecnologia da informação no ambiente educacional pode acontecer de várias formas. Como exemplo, citam-se os aplicativos voltados para o setor administrativo das escolas (software educacional), os aplicativos voltados para a utilização por parte de professores e alunos em sala de aula (software educativo), acesso a sites e vídeos através da internet, jogos etc. No entanto, de acordo com o já apresentado até aqui, os softwares podem ser proprietários ou livres. Sendo assim, os softwares educativos ou os softwares educacionais também podem ter essas características. Posto isso, cabe à escola o papel de detectar e definir que tipo de software será utilizado no ambiente educacional e como este será utilizado. Essa

[24] Tecnologias da Informação e Comunicação (TIC).

definição pode influenciar no tipo e na qualidade da educação que essa escola fornece aos seus alunos.

De acordo com Almi (2009), a educação deveria ter como objetivo a formação de pessoas com a capacidade de filtrar e absorver conhecimento de diversas fontes de forma crítica. Essas pessoas devem ser capazes não só de adquirir conhecimentos prontos fornecidos pelas escolas, mas também de encontrarem suas próprias fontes de conhecimento. A autora afirma que:

> […] mais do que transmitir informação, a educação escolar deve fazer com que o aluno possa reorganizar todo conhecimento construído mediante as várias informações recebidas dos mais diversos meios de comunicação que circulam no mundo globalizado. A escola deve saber filtrar ou saber fazer com que seu educando seja crítico e reflexivo frente às informações que recebe, reorganizando seu pensamento e elaborando novos conhecimentos (ALMI, 2009, p. 11).

Dessa forma, a educação irá contribuir para a emancipação do indivíduo. Os meios pelos quais os alunos adquirem e organizam essas informações podem influenciar no conteúdo final do aprendizado deles, na sua formação como cidadão e na sua inserção na sociedade. Conforme afirma Stallman (2010), a missão social das escolas inclui ensinar os alunos a tornarem-se cidadãos de uma sociedade livre, sociedade esta que, ao mesmo tempo, deve ser forte, independente e cooperativa. Segundo a Constituição da República Federativa do Brasil (1988, p. 123), artigo 205:

> A educação, direito de todos e dever do Estado e da família, será promovida e incentivada com a colaboração da sociedade, visando ao pleno desenvolvimento da pessoa, seu preparo para o exercício da cidadania e sua qualificação para o trabalho.

Promover a produção e a utilização de software livre, principalmente nas escolas, pode contribuir para isso. Portanto, seria importante que essas escolas incentivassem o uso de software livre por parte de alunos e professores. No pensamento de Stallman, esse incentivo deve ser tão importante quanto o da reciclagem. Ele afirma ainda que o dever da escola é ensinar a boa cidadania, trabalhar no aluno o senso comunitário e criar nele o hábito de ajudar os outros. Em computação, isso significa compartilhar conhecimento e, em especial, programas de computador e código-fonte. Para o autor, “[…] ensinar os alunos a usarem software livre e a participarem gratuitamente da comunidade de software é uma lição prática de educação cívica. Também ensinar aos alunos o modelo de papel do serviço

público […]” (STALLMAN, 2010, p. 58), demonstrando na prática como eles devem agir em comunidade.

Stallman acredita também que essa decisão da escola pode ajudar a sociedade como um todo a escapar da dominação tecnológica das megacorporações. Ele sustenta que as escolas devem se recusar a seguir com esse plano de ensino que causa dependência nos alunos. Conforme afirmação do autor apresentada no capítulo 4, “Liberdade e controle sobre software”, na seção 4.8, “Superar a inércia social”, os programas de gratuidade e descontos implantados por essas corporações são uma armadilha e têm a função de formar estudantes viciados em seus produtos (STALLMAN, 2010).

Quando esses alunos se formam e se tornam profissionais, eles vão para o mercado de trabalho, porém os descontos não os acompanham. A partir daí, já como profissionais, eles passam a implantar nas empresas para as quais eles vão trabalhar softwares proprietários, fazendo com que a demanda dessas empresas passe a ser por profissionais que tenham conhecimento desses softwares. Com essa demanda das empresas, as instituições de ensino precisam sempre oferecer treinamentos voltados para esses tipos de softwares. Esse movimento torna-se então um círculo vicioso (STALLMAN, 2010). Porém, quando os estudantes aprendem desde cedo a utilizar software livre, eles têm a tendência de utilizar software livre quando se formam e vão para o mercado de trabalho, quebrando assim esse círculo.

O ensino de softwares livres e de outras disciplinas utilizando softwares livres pode despertar nos alunos o desejo de se aprofundarem mais no conhecimento de computação e de outras disciplinas, não superficialmente, mas de forma mais profunda e com maior qualidade no aprendizado.

Os programas utilizados em educação tecnológica precisam desenvolver o potencial intelectual desses alunos através da possibilidade de poderem ser alterados por eles mesmos para atenderem a demandas específicas. Isso pode fazer com que os alunos passem a conhecer os programas como de fato eles são, ou seja, como são desenvolvidos. Esse conhecimento se dá através do acesso ao código-fonte e do contato com as técnicas de programação. Esse acesso trará para os alunos não só o conhecimento da área específica de estudo, mas também de construção de ferramentas para solução de problemas.

Os benefícios oferecidos com a utilização do software livre não são apenas para os alunos, pois os professores também entrarão em contato com essas ferramentas e passarão a utilizá-las. Para além dos processos de ensino-aprendizagem, a utilização de software livre por parte dos professores poderá desenvolver neles uma nova perspectiva pedagógica, a de autores de recursos pedagógicos digitais. Isso se dá porque eles passam a desenvolver projetos próprios para serem utilizados em sala de aula. “Essa prática autoral do professor contribui para a melhoria da sua autoestima e o motiva para continuar explorando as possibilidades pedagógicas do computador e do software livre em sua prática de ensino” (FERNANDES, 2011, p. 5).

Nessa perspectiva, os professores passam de meros utilizadores das técnicas pedagógicas a produtores delas. Essa impressão é compartilhada por Oliveira, Moura e Sousa (2015, p. 79), quando afirmam que “[…] hoje, diante das tecnologias apresentadas aos alunos, o professor tem o papel de interventor dessa nova forma de ensino, dando o suporte necessário ao uso adequado e responsável dos recursos tecnológicos”. No entanto, para que os professores possam atuar de forma satisfatória e possam usufruir dessas tecnologias, eles precisam estar capacitados e atualizados. Logo, cabe aos professores, com apoio das instituições de ensino e da sociedade privada, uma busca constate do aprofundamento dos conhecimentos que se fazem necessários no exercício das práticas pedagógicas (ZÍLIO, 2013).

Com o software livre, toda a comunidade escolar ganha, pois todos passam de utilizadores para a possibilidade de criadores. A reflexão de Souza (2008, p. 50) reforça essa ideia:

> A utilização de soluções livres nas escolas, especialmente nas públicas, vai muito além do fato de poder economizar *royalties* com a aquisição de licenças, pois alcança a possibilidade de podermos adequá-los às nossas necessidades e ajudarmos a comunidade escolar a participar dessa elaboração.

Essa mudança no papel do professor que passa de usuário para desenvolvedor poderá contribuir para a melhoria da educação em geral, no sentido de que mais pessoas estarão trabalhando no desenvolvimento de melhores ferramentas e práticas pedagógicas.

Além de professores e alunos, a própria educação se beneficia do software livre, pois a sua relação com esse tipo de software não acontece apenas na

disponibilidade de softwares educativos ou educacionais. Essa relação está também ligada à completude que um proporciona ao outro. Rajani, Rekola e Mielonen (2003) explicam que software livre e outros tipos de software com código-fonte aberto *(*Free/Libre Open Source Software – FLOSS) tem uma relação com a educação de forma que os dois podem se completar de modo recíproco. Essa relação é benéfica para a educação, mas também ajuda na difusão desses softwares, de forma que a utilização deles pode ajudar a cumprir o potencial deles.

Ao mesmo tempo, essa utilização pode ajudar também a aumentar e a melhorar a qualidade da educação, oferecendo boas ferramentas para promovê-la. Um bom exemplo é a distribuição Linux Edubuntu, da empresa Canonical. Esse sistema operacional é voltado para ambiente educacional. Essa distribuição conta com diversos softwares educativos, com destaque para o Gcompris e o KDEedu. O Gcompris é "[…] uma coleção de atividades destinadas a crianças de 2 a 10 anos de idade […] e traz atividades de lógica, matemática, leitura, exploração do computador, atividades de percepção visual, sinestésica e auditiva […]", entre outros (LEITÃO, s.d., p. 4). Já o KDEedu "[…] traz 14 aplicativos divididos em línguas, matemática, química, geometria, programação logo e um editor de testes que pode ser usado por alunos ou professores" (LEITÃO, s.d., p. 5). Esse sistema operacional pode ser conseguido gratuitamente pela instituição de ensino no site da Canonical e instalados nos computadores de forma simplificada.

Além de proporcionar ferramentas boas e prontamente disponíveis, muitas vezes de forma gratuita, o software livre ainda fornece exemplos positivos ao redor do mundo, inclusive no Brasil. Por exemplo, o governo brasileiro criou um projeto chamado Linux Educacional. Esse projeto busca o melhor aproveitamento dos ambientes de informática já disponíveis nas escolas. Isso acontece através da utilização de uma distribuição Linux que leva o mesmo nome do projeto, nos laboratórios de informática dessas escolas. Essa distribuição oferece aos professores e alunos diversas ferramentas voltadas ao ambiente educacional, possibilitando uma interação maior e a possibilidade de, juntos, criarem seus próprios processos de ensino-aprendizagem (LINUX EDUCACIONAL, 2017).

No ano de 2009 o Linux Educacional foi utilizado na cidade de Fortaleza nas escolas públicas municipais. Um estudo feito pela professora Jaiza Helena Moisés Fernandes, em seis turmas do segundo ano do ensino fundamental, totalizando 150 alunos entre sete e oito anos de idade e seus respectivos professores, mostrou a

importância desse projeto. Além do aumento do interesse nas atividades propostas por parte dos alunos, eles se empenharam mais para concluí-las com êxito. Além disso, eles “[…] relatavam que estavam felizes, pois aprenderam a ler e a escrever através do computador” (FERNANDES, 2011, p. 13). Outro resultado positivo da adoção do Linux Educacional foi uma melhora na nota geral do Índice de Desenvolvimento da Educação Básica (IDEB).

Esses resultados obtidos no estudo em Fortaleza reforçam a reflexão que Rajani, Rekola e Mielonen (2003) fazem sobre o fato de a maioria dos softwares livres estarem prontamente disponíveis e sobre a liberdade de leitura do código-fonte. Eles afirmam que isso os tornam ainda mais importantes, pois, além da facilidade no desenvolvimento de novas ferramentas educacionais, o software livre proporciona a possibilidade de facilitar a extração das melhores funções das ferramentas já existentes.

Outro fator importante a ser mencionado é a barreira do idioma. Existe uma grande dificuldade para a aquisição de softwares educativos por pessoas que falam um idioma pouco conhecido, pois a maioria dos softwares são desenvolvidos em países com os idiomas mais falados no mundo, como é o caso do inglês, francês, espanhol e português. Isso ocorre porque não há interesse por parte das grandes corporações em desenvolvê-los, devido à baixa quantidade de usuário (RAJANI; REKOLA; MIELONEN, 2003). Sendo assim, trocar a utilização de software proprietário por software livre pode ser a solução para esse problema, pois as comunidades locais podem desenvolver programas e traduções para os mais diversos idiomas, tais como o Svenska ou o Zulu.

A dificuldade de acesso à informação nas escolas e universidades em países em desenvolvimento não se dá apenas pela ausência de softwares educativos e educacionais, mas também pela insistência dessas instituições em utilizarem softwares proprietários nos ambientes acadêmicos. Essa utilização faz com que a informação sobre o funcionamento desses softwares fique restrita aos seus desenvolvedores. Portanto, principalmente as universidades, precisam fomentar o uso e a pesquisa sobre software livre para que a informação e o conhecimento sejam difundidos. As pesquisas universitárias poderiam ser focadas no interesse público. Da mesma forma que as universidades mantêm um incentivo a seus pesquisadores das diversas áreas do conhecimento, elas poderiam incentivar o

software livre, pois criar software livre também contribui para o conhecimento humano (STALLMAN, 2010).

Na visão de Carvalho (2015, p. 12):

> A universidade como instituição de ensino, precisa transmitir as informações para o desenvolvimento do conhecimento e, pela extensão, um dos aspectos do tripé da educação[25] em que se baseiam as instituições públicas, tem o papel de desenvolver o indivíduo como cidadão partícipe dos problemas presentes no dia a dia não apenas reprodutor de modelos aprendidos.

Com a adoção, a difusão e o desenvolvimento do software livre nas escolas e universidades, esses cidadãos poderão participar do desenvolvimento de soluções para os problemas da sociedade em que se encontram, contribuindo assim para o desenvolvimento dela.

A utilização de tecnologias na educação apresenta vantagens e desvantagens, e não faltam críticos dessa utilização. Neste sentido, Cruz *et al*. (s.d., p. 2) afirmam que "essa explosão tecnológica também acarreta em problemas relacionados com o despreparo dos usuários. […] Com a introdução da tecnologia na educação, esses problemas começam a refletir na formação social, educacional e profissional." No entanto, são inúmeras as vantagens da utilização de tecnologias em educação, mas elas são ampliadas quando são utilizadas tecnologias da informação e principalmente softwares livres, pois isso propicia aos estudantes e professores a liberdade de escolha e a possibilidade de aprenderem mais e melhor, fazendo com que eles possam participar do desenvolvimento dos instrumentos tecnológicos utilizados em sala de aula. Além disso, essa experiência do aluno com uma ferramenta na qual ele tem o controle e que permite a ele alterá-la para satisfazer suas necessidades possibilita o aprendizado de resolução de problemas, além de incutir nele o senso de comunidade.

[25] Ensino, pesquisa e extensão.

6 CONSIDERAÇÕES FINAIS

O objetivo geral desta pesquisa foi compreender as teses do fundador do GNU Project e da Free Software Foundation, Richard Stallman, referentes à filosofia do software livre e relacioná-la à educação profissional e tecnológica.

Na literatura pesquisada, o software livre é tratado de forma prática, isto é, com estudos de casos de implementações desse tipo de software em ambiente educacional. As discussões se deram em torno da utilização do mesmo por professores e alunos, da adaptabilidade desses aos softwares e de questões práticas relacionadas à implantação e aceitação. Não foi possível encontrar, na pesquisa realizada, literatura que aproximasse o tema do software livre e da educação de forma que fizesse uma relação entre os dois.

Para atingir os objetivos delimitados, foi realizada uma pesquisa teórica sobre a obra de Richard Stallman com a técnica de análise de conteúdo. Esse trabalho foi organizado em quatro capítulos principais: no capítulo 2, “A revolução digital: seus personagens, empresas e acontecimentos históricos”, foram apresentados os personagens, as empresas e os acontecimentos históricos que levaram ao surgimento da revolução digital, com a criação dos computadores pessoais, das licenças de software e dos softwares livres e qual a relação entre eles. Esse capítulo buscou compreender o contexto em que o software livre se desenvolveu e é um ponto de partida para a compreensão da importância do software em ambiente educacional.

No capítulo 3, intitulado “Software livre e software proprietário: definições iniciais” destacaram-se os modelos de desenvolvimento e comercialização de softwares e as questões técnicas e práticas a respeito de cada um dos modelos. Além disso, foram expostas algumas terminologias e definições necessárias para compreensão da discussão. Esse capítulo apresentou ao leitor as questões técnicas que podem influenciar na escolha de um software. No capítulo 4, denominado “Liberdade e controle sobre software”, foram demonstrados os fundamentos da filosofia do software livre de Richard Stallman, suas implicações técnicas e filosóficas. Foi discutida também a questão sobre o controle do desenvolvimento de software e a liberdade do seu uso. Esse capítulo é fundamental para a compreensão das ideias do autor sobre liberdade de escolha, de criação e de disseminação de conhecimento. Por fim, no capítulo 5, intitulado “Contribuições do software livre para

educação: educação cívica e emancipação do indivíduo", foram discutidas as contribuições que o software livre pode oferecer para a educação e para o ambiente educacional. Nesse capítulo foram apresentados importantes conceitos que ajudam na discussão do uso de tecnologia em ambiente educacional, seus possíveis impactos e sua contribuição para uma educação emancipadora.

A pesquisa apresenta como resultado a ideia de que a utilização do software livre tende a contribuir para uma educação emancipadora, ou seja, valoriza o livre pensamento. Isso evidencia que a utilização de software livre pode conduzir o indivíduo a uma educação cívica em que este tenha consciência de seu papel como parte de uma sociedade. Essa contribuição acontece pelo fato de o software livre permitir ao usuário o estudo do código-fonte. Isso faz com que seja possível estudá-lo, modificá-lo e distribuí-lo para que esse software cumpra seu papel social. Tanto alunos quanto professores se beneficiam da liberdade proporcionada pelo software livre, na medida em que o aluno passa a ter a possibilidade de conhecer melhor as ferramentas que são utilizadas em sala de aula e o professor passa de mero utilizador para a condição de possível criador de ferramentas educacionais.

O papel do software livre no ambiente educacional é o de libertar esse ambiente das amarras ideológicas das grandes corporações, que visam ao controle e ao lucro acima de tudo, tanto educacionais quanto tecnológicas, através do incentivo ao questionamento do que está sendo entregue aos seus utilizadores e da possibilidade desses recriarem esse ambiente com suas próprias ideias.

Acredita-se que o objetivo geral desta pesquisa tenha sido atingido. Isso porque, ao longo do trabalho, foi possível mapear e reunir, em uma visão conjunta, as questões técnicas, éticas, ideológicas e pedagógicas na obra do autor. Isso se deu através do alcance dos objetivos específicos trabalhados em cada um dos capítulos. Sendo assim, espera-se que a questão inicial, "Que relações podem ser pensadas entre a filosofia do software livre de Richard Stallman e a educação profissional e tecnológica?", tenha sido respondida, na medida em que se compreende que a filosofia de software livre está relacionada à liberdade de escolha individual e à liberdade de pensamento. E essa filosofia prega que cada um deve ter a possibilidade de ter controle das suas ideias e deve poder criticar tanto as ferramentas quanto as informações que recebe. Além disso, a filosofia de software livre afirma que a educação deve ser instrumento de emancipação cultural e deve formar cidadãos preocupados com questões éticas, cívicas e sociais. Isso vai ao

encontro do que foi apresentado no capítulo em que se verificou que a educação deveria ter como objetivo a formação de pessoas com a capacidade de filtrar e absorver conhecimento de diversas fontes de forma crítica. Essas pessoas devem ser capazes não só de adquirir conhecimentos prontos fornecidos pelas escolas, mas também de encontrarem suas próprias fontes de conhecimento.

A pesquisa realizada concentrou-se em questões teóricas relacionadas à utilização de software em ambiente educacional. Portanto, não foram mapeadas questões práticas relacionadas a projetos pedagógicos específicos utilizando-se software livre em ambiente educacional. Para uma compreensão ainda mais global do pensamento de Stallman, é importante que seja feita um estudo empírico a respeito do tema para verificar na prática os conceitos aqui estudados.

REFERÊNCIAS

AGUADO, A. G. **O movimento do software livre e suas contribuições para a formação de redes de colaboração na educação**. 2012. 139 p. Dissertação (Mestrado) – Faculdade de Tecnologia, Universidade Estadual de Campinas, Limeira-SP.

AGUIAR, D. C. **Introdução ao desenvolvimento de software embarcado**. Natal: Sociedade Brasileira de Computação, 2011.

ALENCAR, A. F. **A pedagogia da migração do software proprietário para o livre**: uma perspectiva freiriana. 2007. Dissertação (Mestrado em Educação) – Faculdade de Educação, Universidade de São Paulo, São Paulo, 2007.

ANDERSON, R. **Cryptography and competition policy**: issues with 'Trusted Computing'. Cambrigde: Cambrigde University, 2002.

ANDRADE, W. L. S. de. **Aprendizagem mediada por tecnologias digitais baseadas em software livre no âmbito do programa Um Computador por Aluno – PROUCA**. 2013. 172 f. Dissertação (Mestrado em Educação Matemática e Tecnológica) – Centro de Educação, Universidade Federal de Pernambuco, Recife, 2013.

ARAGÃO, D. W. **Macuco**: dez anos de blogagem. Curitiba: Clube dos Autores, 2015.

BARDIN, L. **Análise de conteúdo**. Lisboa: Edições 70, 1977.

BARIFOUSE, R. No mundo de Woz. **Época Negócios**, [s.d.]. Disponível em: <http://epocanegocios.globo.com/Revista/Epocanegocios/0,,EDR84489-15918,00.html>. Acesso em: 24 ago. 2018.

BETZ, D.; EDWARDS, J. **Entrevista à BYTE com Richard Stallman**. Richard Stallman fala sobre seu sistema de software compatível com Unix de domínio público com editores da BYTE (julho de 1986). Disponível em: <https://www.gnu.org/gnu/byte-interview.html>. Acesso em: 28 nov. 2017.

BEZERRA, R. M. **Sistemas operacionais**. Salvador: CEFET-BA, 2008. [Notas de aulas].

BLOG TIQX. **Linux e interfaces gráficas de usuário**: um leque repleto de opções! 29 abr. 2015. Disponível em: <http://tiqx.blogspot.com.br/2015/04/linux-e-interfaces-graficas-de-usuario_29.html>. Acesso em: 30 mar. 2018.

BRANDÃO, M. Da arte do ofício à ciência da indústria: a conformação do capitalismo industrial no Brasil vista através da educação profissional. **Boletim Técnico do SENAC**, v. 25, n. 3, p. 17-30, set./dez. 1999.

BRASIL. Decreto nº 7.566, de 23 de setembro de 1909. Cria nas capitais dos Estados da Republica Escolas de Aprendizes Artifices, para o ensino profissional primario e gratuito. **Diário Official**, 26 set. 1909, p. 6975 (Publicação Original). Disponível em: <https://www2.camara.leg.br/legin/fed/decret/1900-1909/decreto-7566-23-setembro-1909-525411-publicacaooriginal-1-pe.html>. Acesso em: 10 maio 2019.

BRASIL. Decreto nº 9.609, de 19 de Fevereiro de 1998. Dispõe sobre a proteção da propriedade intelectual de programa de computador, sua comercialização no País, e dá outras providências. **Diário Official**, 20 fev. 1998, p. 1 (Publicação Original). Disponível em: < http://www.planalto.gov.br/ccivil_03/leis/l9609.htm>. Acesso em: 24 ago. 2019.

BRASIL. Constituição (1988). **Constituição da República Federativa do Brasil**, 1988. Brasília: Senado Federal, Centro Gráfico, 1988.

CAMPOS, C. J. G. Método de análise de conteúdo: ferramenta para a análise de dados qualitativos no campo da saúde. **Rev. Bras. Enferm**., Brasília, p. 611-614, 2004.

CARVALHO, Ricardo Ferreira de. **Políticas públicas para o software livre na educação superior**: o uso do programa Scribus no curso de Jornalismo da Universidade Federal de Uberlândia. 2015. 86 f. Dissertação (Mestrado em Ciências Humanas) – Universidade Federal de Uberlândia, Uberlândia, 2015.

CAVALINI, M. **Pesquisa teórica e pesquisa empírica**. 2016. Dispónivel em: <http://www.midia.uff.br/metodologia/?p=169694>. Acesso em: 3 fev. 2019.

CASTRO, P. D. **Serviço de informação ao cidadão**. [s.d.]. Disponível em: <http://www.ibict.br/servico-de-informacao-ao-cidadao-1/Relatoriofinal_PriscilaPaiva_anexo_BigData.pdf>. Acesso em: 16 out. 2018.

CERON, J. M.; GRANVILLE, L.; TAROUCO, L. Taxonomia de *malwares*: uma avaliação dos *malwares* automaticamente propagados na rede. In: SIMPÓSIO BRASILEIRO EM SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS, 9., 2009, Porto Alegre. **Anais…** Porto Alegre: Universidade Federal do Rio Grande do Sul, 2009.

COLANGELO, L. F. **Implantação de sistemas ERP (Enterprise Resource Planning)**: um enfoque de longo prazo. São Paulo: Atlas, 2001.

COMPUTERWORLD. **Windows 7 ainda é o sistema operacional mais usado no mundo**. 3 ago. 2017. Disponível em: <http://computerworld.com.br/windows-7-ainda-e-o-sistema-operacional-mais-usado-no-mundo>. Acesso em: 18 jul. 2018.

CORTE CERTO. **Hobbysta, hobista, hobbista, hobbyista**. 16 jan. 2015. Disponível em: <https://cortecerto.com/glossario/hobbysta/>. Acesso em: 9 jan. 2019.

CORTES, R. L. **Ações coletivas com uso da internet**: o caso do Projeto Jogo Justo. 2013. 135 f. Dissertação (Mestrado em Educação) – Faculdade de Educação, Universidade de São Paulo, São Paulo, 2013.

CRUZ, V. M. *et al*. **Informática e educação**: pontos negativos. Computadores e Sociedade I. São Paulo: Universidade de São Paulo, [s.d.]. Disponível em: <http://wiki.icmc.usp.br/images/4/43/Inform%C3%A1tica_e_Educa%C3%A7%C3%A3o_%E2%80%93_Pontos_Negativos.pdf>. Acesso em: 10 maio 2019.

DARÓS, R. C. **Software livre e educação**. 2010. 79 f. Dissertação (Mestrado em Educação nas Ciências) – Departamento de Pedagogia, Universidade Regional do Noroeste do Estado do Rio Grande do Sul, Ijuí, 2010.

DEMO, P. **Metodologia do conhecimento científico**. São Paulo: Atlas, 2000.

DINIZ, S. N. de F. **O uso das novas tecnologias em sala de aula**. 2001. 173 f. Dissertação (Mestrado em Engenharia de Produção) – Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis, 2001.

FERNANDES, J. H. M. (2011). *Software livre* na educação para além da inclusão digital e social: letramentos múltiplos de professores e alunos. **Texto Livre: Linguagem e Tecnologia**, v. 4, n. 1, p. 2-15, out. 2011.

FISL – Fórum Internacional de Software Livre. 26 set. 2014. Disponível em: <https://softwarelivre.org/fisl>. Acesso em: 1 jun. 2018.

FREE SOFTWARE FOUNDATION. **Diretrizes para distribuições de sistemas livres (GNU FSDG)**. 2018. Disponível em: <http://www.gnu.org/distros/free-system-distribution-guidelines.html>. Acesso em: 27 mar. 2018.

FREITAS, D. N. **Metodologia de desenvolvimento de software livre com arquiteturas orientadas a serviços**: um estudo de caso em um ambiente de tradução automática. 2009. 79 f. Dissertação (Mestrado em Informática) – Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná, Curitiba, 2009.

Freire Paulo. **À sombra desta mangueira**. São Paulo: Olho d'Água, 1995.

FREIRE, P. Educação: o sonho possível. In: BRANDÃO, C. R. (Org.). **O educador**: vida e morte. Rio de Janeiro: Graal, 1982. p. 89-101.

FREIRE, P. **Educação como prática de liberdade***.* Rio de Janeiro: Paz e Terra, 1967.

FRIGOTTO, G. A relação da educação profissional e tecnológica com a universalização da educação básica. **Educ. Soc**., Campinas, v. 28, n. 100 - Especial, p. 1129-1152, out. 2007.

FUMEC – Fundação Mineira de Educação e Cultura. **Curso Superior de Tecnologia-em Redes de Computadores**. 2017. Disponível em: <http://vestibular.fumec.br/wp-content/uploads/2018/01/superior-de-tecnologia-em-redes-de-computadores.pdf>. Acesso em: 30 jul. 2018.

GALVÃO, M. A. **SIMULEGO**: um ambiente de simulação para robótica educacional. 2014. 112 f. Dissertação (Mestrado Integrado Profissional em Computação Aplicada) – Instituto Federal de Educação, Universidade Estadual do Ceará, Fortaleza, 2014.

GARCIA, M. N. *et al*. Software livre em relação ao software proprietário: aspectos favoráveis e desfavoráveis percebidos por especialistas. **Gestão & Regionalidade**, v. 26, n. 78, p. 106-120, jan. 2011.

GATES, B.; MYHRVOLD, N.; RINEARSON, P. **A estrada do futuro**. São Paulo: Companhia das Letras, 1995.

G1. Software livre é usado em 48% das microempresas no Brasil, diz IBGE. **G1**, Tecnologias e Games, 13 dez. 2012. Disponível em: <http://g1.globo.com/tecnologia/noticia/2012/12/software-livre-e-usado-em-48-das-microempresas-no-brasil-diz-ibge.html>. Acesso em: 21 maio 2018.

G1. Zuckerberg reconhece que erros no Facebook permitiram roubo de dados. **G1**, Jornal Nacional, 21 mar. 2018. Disponível em: <http://g1.globo.com/jornal-nacional/noticia/2018/03/zuckerberg-reconhece-que-erros-no-facebook-permitiram-roubo-de-dados.html>. Acesso em: 8 jun. 2018.

GNU.Org. **Software livre e educação**. 2011. Disponível em: <https://www.gnu.org/education/education.pt-br.html>. Acesso em: 20 jan. 2019.

GOMES, M. F.; NOVAES, R. V.;BECKE, M. G. Software livre, licenciamento de software. **Nomos – Revista do Programa de Pós-Graduação em Direito**, Universidade Federal do Ceará, Fortaleza, v. 36.2, p. 307-323, 2016.

GOMES, R. O. de A. **Aprendizagem e ensino com software livre**: pesquisa intervenção na formação de professores. 2007. 164 f. Dissertação (Mestrado Acadêmico em Educação) – Centro de Educação, Universidade Estadual do Ceará, Fortaleza, 2007.

GONÇALVES, J. **DNS – Domain Name System**. Departamento de Informática, Universidade Federal do Espírito Santo, Vitória, 2015. Slide. Disponível em: <https://www.inf.ufes.br/~zegonc/material/Redes_de_Computadores/Dns01-ufes.pdf>. Acesso em: 10 maio 2019.

GROSS, M. **Richard Stallman**: High School misfit, symbol of free software, MacArthur-Certified Genius. Interviewed in New York City and Cambridge, Mass., in early 1999. Disponível em: <http://mgross.com/writing/books/my-generation/bonus-chapters/richard-stallman-high-school-misfit-symbol-of-free-software-macarthur-certified-genius/>. Acesso em: 28 nov. 2017.

HELPA, J. P. **Curso I – Fundamentos Conceitos e Práticas da EP**. Curitiba: Associação de Escolas Cristãs de Educação por Princípios, 2015. Trabalho apresentado conforme exigência do programa de EAD. Curso 1 – Fundamentos Conceitos e Práticas da EP. Disponível em: <http://www.atuacaovoluntaria.org.br/upload/juliana-pompeo-helpa-conceito-de-educacao-ead-aecep_3.pdf>. Acesso em: 11 maio 2019.

HESSEN, J. **Teoria do conhecimento**. São Paulo: Martins Fontes, 2000.

IHU – Instituto Humanitas Unisinos. **Monitoramento digital é intolerável, afirma Stallman**. 2013. Disponível em: <http://www.ihu.unisinos.br/noticias/521763-monitoramento-digital-e-intoleravel-afirma-stallman>. Acesso em: 14 maio 2019.

ISAACSON, W. **Os inovadores**: uma biografia da revolução digital. Tradução de Berilo Vargas, Luciano Vieira Machado e Pedro Maia Soares. São Paulo: Companhia das Letras, 2014.

JAEGER, W. **Paideia**: a formação do homem grego. Tradução de Artur M. Parreira. São Paulo: Martins Fontes, 1986.

JONES, K. C. A rare glimpse into Richard Stallman's world. **InformationWeek**, 1 jun. 2006. Disponível em: <https://www.informationweek.com/a-rare-glimpse-into-richard-stallmans-world/d/d-id/1039353>. Acesso em: 28 nov. 2017.

JUCA, S. C. A relevância dos softwares educativos na educação profissional. **Ciências & Cognição**, v. 8, p. 22-28, 2006.

KANT, E. **Crítica da razão pura**. [S.l.]: Acrópolis, 2001.

KAHNEY, L. **A cabeça de Steve Jobs**. Rio de Janeiro: Agir Editora, 2008.

KANTOWITZ, B. H.; ROEDIGER III, L. H.; ELMES, D. G. **Psicologia experimental**: psicologia para compreender a pesquisa em psicologia. São Paulo: Cengage, 2006.

KROPIWIEC, D. D. **Trusted Computing**: questões técnicas e sociais e suas controvérsias. Unicamp, Campinas, 2005. Trabalho apresentado a disciplina de

MO826. Dispónivel em: <http://www.ic.unicamp.br/~rdahab/cursos/mo826/2005/trabalhos/DiogoKropiwiec_TrustedComputing.pdf>. Acesso em: 11 maio 2019.

LEONARD, D. Hey, PC, Who Taught You to Fight Back? **The New York Times**, Media, 29 ago. 2009. Disponível em: <https://www.nytimes.com/2009/08/30/business/media/30ad.html>. Acesso em: 24 out. 2018.

LESSA, A. L. **Os aspectos jurídicos e econômicos da pirataria no Brasil vistos pelo ângulo das relações internacionais**. Monografia (Graduação em Ciência da Computação) – Centro Universitário de Brasília, Brasília, 2003.

LEITÃO, A. A. P. **Edubuntu**: um Linux voltado à educação digital. [s.d.]. Dispónivel em: <http://www.hipertextus.net/volume1/artigo3-andre-padilha.pdf>. Acesso em: 11 maio 2019.

LINUX's History. Disponível em: <https://www.cs.cmu.edu/~awb/linux.history.html>. Acesso em: 13 maio 2019.

LINUXPLACE. **Linux Place – Empresa**. 2008. Disponível em: <http://linuxplace.com.br/institucional>. Acesso em: 22 maio 2018.

LOMBARDI, J. C. A revolução russa e os desafios à pedagogia histórico-crítica. **Germinal: Marxismo e Educação em Debate**, Salvador, v. 9, n. 3, p. 292-306, dez. 2017.

ALMI, M. M. **Função social da escola e o projeto pedagógico**. 2009. 43 f. Monografia (Especialização em Gestão Educacional) – Centreo de Educação, Curso de Pós-Graduação a Distância em Educação, Universidade Federal de Santa Maria, Constantina-RS, 2009.

TREND MICRO. **Contrato de licença**. [S.l.]: Trend Micro, Incorporated, set. 2013.

LINUX EDUCACIONAL. **Sobre**. 2017. Disponível em: <https://linuxeducacional.c3sl.ufpr.br/sobre/>. Acesso em: 19 nov. 2018.

MICROSOFT. **O suporte estendido ao Windows Server 2003 terminou em 14 de julho de 2015**. 2015. Disponível em: <https://www.microsoft.com/pt-br/cloud-platform/windows-server-2003>. Acesso em: 21 maio 2018.

MICROSOFT. **Microsoft contribui para o ecossistema aberto ao se juntar à Linux Foundation e dar boas-vindas ao Google na comunidade.NET**. 16 nov. 2016. Disponível em: <https://news.microsoft.com/pt-br/microsoft-contribui-para-o-ecossistema-aberto-ao-se-juntar-a-linux-foundation-e-dar-boas-vindas-ao-google-na-comunidade-net/>. Acesso em: 29 abr. 2018.

MICROSOFT. **Office**. 2018. Disponível em: <https://products.office.com/pt-br/student/office-in-education#FAQs>. Acesso em: 18 out. 2018.

MIT. **Frequently Asked Questions**. 2007. Disponível em: <http://hacks.mit.edu/Hacks/misc/faq.html>. Acesso em: 13 maio 2019.

MEC – Ministério da Educação. Resolução CNE/CP nº 3, de 18 de dezembro de 2002. Institui as Diretrizes Curriculares Nacionais Gerais para a organização e o funcionamento dos cursos superiores de tecnologia. **Diário Oficial da União**, Brasília, 23 dez. 2002, p. 465-467.

MORAES, C. H. de; KOHN, K. O impacto das novas tecnologias na sociedade: conceitos e características da Sociedade da Informação e da Sociedade Digital. In: CONGRESSO BRASILEIRO DE CIÊNCIAS DA COMUNICAÇÃO, 30., 2007, Santos. **Anais…** Santos: Intercom – Sociedade Brasileira de Estudos Interdisciplinares da Comunicação, 2007.

MORAIS, R. X. T. **Software educacional**: a importância de sua avaliação e do seu uso nas salas de aula. 2003. 52 f. Monografia (Bacharelado em Ciência da Computação) – Faculdade Lourenço Filho, Fortaleza, 2003.

MOURA, E. L.; MOREIRA, M. A. **Metodologias de desenvolvimento de software**.Uberlândia: Faculdade Pitágoras, 2012. Trabalho de Especialização em Desenvolvimento de Aplicativos Web.

MOURA, J. dos S. **Caminhos pela liberdade do conhecimento**: software livre no assentamento Moacir Wanderley - Quissamã (SE). 2007. 156 f. Dissertação (Mestrado em Educação) – Universidade Federal de Sergipe, São Cristóvão, 2007.

OLIVEIRA, C. C. **Ambientes informatizados de aprendizagem**: produção e avaliação de software educativo. Campinas: Papirus, 2001.

OLIVEIRA, C. de; MOURA, S. P.; SOUSA, E. R. TIC’s na educação: a utilização das tecnologias da informação e comunicação na aprendizagem do aluno. **Pedagogia em Ação**,v. 7, n. 1, p. 75-95, 2015.

OLIVEIRA, J. V.; FERES, M. V.; GONÇALVE, D. D. Robin Hood às avessas: software, pirataria e direito autoral. **Rev. Direito GV**, São Paulo, v. 13, n. 1, p. 69-94, abr. 2017.

OLIVEIRA, L. X. de. **Política de formação de professores e inclusão digital**: o uso do software livre. 2008. 203 f. Dissertação (Mestrado Acadêmico em Educação) – Centro de Educação, Universidade Estadual do Ceará, Fortaleza, 2008.

OPERATING SYSTEM Market Share. **Net Marketshare**. 31 jul. 2018. Disponível em <https://www.netmarketshare.com/>. Acesso em: 31 jul. 2018.

PC WORLD. The 7 worst tech predictions of all time. **PC WORLD**, 31 dez. 2008. Disponível em: <https://www.pcworld.com/article/155984/worst_tech_predictions.html>. Acesso em: 10 jan. 2019.

PIAGET, J. **Para onde vai a educação?** 8. ed. Rio de Janeiro: José Olimpyo, 1984.

PERANI, L. Computadores para o povo: games e hobbyismo nas revistas especializadas em computação. In: CONGRESSO BRASILEIRO DE CIÊNCIAS DA COMUNICAÇÃO, 39., 2016, São Paulo. **Anais…** São Paulo: Intercom – Sociedade Brasileira de Estudos Interdisciplinares da Comunicação, 2016.

PORTAL ERP. **Entenda o que é ERP (Sistemas de Gestão Empresarial)**. 30 jan. 2012. Disponível em: <https://portalerp.com/entenda-erp-2>. Acesso em: 25 mar. 2018.

PORTAL DA EDUCAÇÃO. **O currículo escolar**. 2018. Disponível em: <https://www.portaleducacao.com.br/conteudo/artigos/educacao/o-curriculo-escolar/18081>. Acesso em: 7 jan. 2019.

RAJANI, N.; REKOLA, J.; MIELONEN, T. **Free as in education**. Helsinki, Finland: Ministry for Foreign Affairs, 2003.

RAYMOND, E. **The Cathedral and the Bazaar**. Sebastopol, CA: O'Reilly Media, 1999.

RONZANI, R. Y. **Entre vilões e mocinhos**: o software livre no contexto das Américas. 2011. Dissertação (Mestrado em História Social) – Faculdade de Filosofia, Letras e Ciências Humanas, Universidade de São Paulo, São Paulo, 2011.

SANTOS, B. P.; SILVA, L. A.; CELES, C. S. **Internet das coisas**: da teoria à prática. Belo Horizonte: Universidade Federal de Minas Gerais, 2017. Minicursos SBRC – Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos. Dispónivel em: <https://homepages.dcc.ufmg.br/~mmvieira/cc/papers/internet-das-coisas.pdf>. Acesso em: 11 maio 2019.

SANTOS, H. J. dos. Educação e ideologia. **Perspectivas**, São Paulo, n. 3, p. 19-28, 1980.

SEBRAE. **Propriedade Intelectual. Definição de Patente**. 19 maio 2017. Disponível em: <http://www.sebrae.com.br/sites/PortalSebrae/artigos/definicao-de-patente,230a634e2ca62410VgnVCM100000b272010aRCRD>. Acesso em: 22 maio 2018.

SILVA, H. *et al*. Inclusão digital e educação para a competência informacional: uma questão de ética e cidadania. **Ci. Inf**., Brasília, v. 34, n. 1, p. 28-36, jan./abr. 2005.

SILVA, J. D. **Aceitação de softwares livres em ambientes de ensino básico**: estudo das escolas públicas do município de Belo Horizonte. 2012. 33 f. Projeto de Pesquisa (Mestrado em Sistema de Informação e Gestão do Conhecimento) – Programa de Pós-Graduação em Sistema de informação, Universidade FUMEC, Belo Horizonte, 2012.

SILVEIRA, S. A. **Software livre**: a luta pela liberdade do conhecimento. São Paulo: Fundação Perseu Abramo, 2004.

SMITH, Daniel. **Pensar como Bill Gates: empreendedor, líder, filantropo** – Uma biografia inspiradora. Amadora: Editora Vogais, 2016.

SOFFA, M. M.; ALCÂNTARA, P. R. O uso do software educativo: reflexões da prática docente na sala informatizada. In: CONGRESSO NACIONAL DE EDUCAÇÃO (EDUCERE), 2008. **Anais…** [s.n.t].

SOFTWARELIVRE. **Windows tem backdoor para NSA desde 1999 enquanto Debian é livre**. 21 de Julho de 2013. Disponível em: <http://softwarelivre.org/portal/noticias/windows-tem-backdoor-para-nsa-desde-1999-enquanto-debian-e-livre>. Acesso em: 6 jun. 2018.

SOLA, J. E. **A proteção dos direitos autorais a partir da realidade internet**: a perspectiva brasileira. 2002. 172 f. Dissertação (Mestrado em Ciência da Informação) – Faculdade de Filosofia e Ciências, Universidade Estual Paulista, Marília, 2002.

SOUZA, C. G. **Formação continuada de professores**: incentivando a utilização do software livre Gcompris em sala de aula. 2016. 134 f. Dissertação (Mestrado) – Curso de Ensino, Universidade do Vale do Taquari, Lajeado, 2016.

SOUZA, G. M. de O. **Navegar é preciso**: viagem nas políticas de adoção do software livre nas escolas públicas municipais de Fortaleza. 2008. 200 f.Dissertação (Mestrado Acadêmico em Educação) – Centro de Educação, Universidade Estadual do Ceará, Fortaleza, 2008.

SOUZA, P. F. **Uso de software livre na capacitação de professores para utilização das TICs**: estudo sobre a aceitação do sistema Muriqui Linux. 2008. Mestrado (Dissertação em Informática) – Pontíficia Universidade Católica de Minas Gerais, Belo Horizonte, 2008.

STALLMAN, R. M. **Software libre para una sociedad libre**. Madrid: Traficantes de Sueños, 2004.

STALLMAN, R. M. **Free software, free society**. Boston: Free Software Foundantion, 2010.

STALLMAN, R. M. **Conferencia de Richard Stallman en español**. 21 jul. 2013. Disponível em: <https://www.youtube.com/watch?v=5t_EcPTEzh4>. Acesso em: 15 jun. 2018.

STALLMAN, R. M. **On Hacking**. Jan. 2017. Disponível em: <https://stallman.org/articles/on-hacking.htm>. Acesso em: 7 jan. 2017.

NYBO, F. E.; LIPO. H. S. **Patente de Software é possível no Brasil?.** 15 abr. 2016. Disponível em: < https://startupi.com.br/2016/04/patente-de-software-e-possivel-no-brasil/>. Acesso em: 24 de ago. 2019.

TAVARES, L. E. **Apropriabilidade, mecanismos de apropriabilidade e inovação no setor de software livre**. 2007. 110 f. Dissertação (Mestrado em Administração) – Centro de Estudos Sociais Aplicados, Universidade Estadual do Ceará, Fortaleza, 2007.

TECHTUDO. **Usuários acusam Microsoft e Lenovo de bloquear Linux no Yoga; entenda**. 22 set. 2016. Disponível em: <http://www.techtudo.com.br/noticias/noticia/2016/09/usuarios-acusam-microsoft-e-lenovo-de-bloquear-linux-no-yoga-entenda.html>. Acesso em: 22 maio 2018.

TECMUNDO. **O que é boot?**. 1 jun. 2012. Disponível em: <https://www.tecmundo.com.br/o-que-e/955-o-que-e-boot-.htm>. Acesso em: 21 mar. 2018.

TORVALDS, L.; DIAMOND, D. **Só por prazer**: os bastidores da sua criação. São Paulo: Campus/Elsevier, 2001.

TSIaD – Tecnologia em Sistemas para Internet à Distância. **1 Sistemas computacionais**. [s.d.]. Disponível em: <http://uab.ifsul.edu.br/tsiad/conteudo/modulo1/sop/ua/at2/>. Acesso em: 20 mar. 2018.

UNIARA. **MBA em Administração de Redes Linux**. 2018. Disponível em: <http://www.uniara.com.br/cursos/ead/pos-graduacao/mba-em-administracao-de-redes-linux/>. Acesso em: 30 jul. 2018.

UOL. Tecnologia. **Apple admite deixar iPhone mais lento conforme bateria vai se deteriorando. 21 dez. 2017**. Disponível em: <https://tecnologia.uol.com.br/noticias/redacao/2017/12/21/apple-admite-deixar-iphone-mais-lento-conforme-bateria-vai-se-deteriorando.htm>. Acesso em: 27 jan. 2018.

UOL. Mercado. **Ministério Público processa Microsoft por coleta de dados no Windows 10**. 25 abr. 2018 Ddisponível em: <https://www1.folha.uol.com.br/mercado/2018/04/ministerio-publico-processa-microsoft-por-coleta-de-dados-no-windows-10.shtml>. Acesso em: 30 abr. 2018.

VASCONCELLOS, C. S. **Planejamento**: plano de ensino-aprendizagem e projeto educativo. São Paulo: Libertat, 1995.

VENTURA, F. **O problema dos e-books**. 8 mar. 2016. Disponível em: <http://gizmodo.uol.com.br/o-problema-dos-e-books/>. Acesso em: 27 jan. 2018.

JAEGER, W.. **Paideia**: a formação do homem grego. Tradução de Artur M. Parreira. São Paulo: Martins Fontes, 1995.

WIKIPÉDIA. **Whole Earth Catalog**. 2017. Disponível em: <https://pt.wikipedia.org/wiki/Whole_Earth_Catalog>. Acesso em: 25 jan. 2019.

WIKIPÉDIA. **BIOS**. 2019. Disponível em: <https://pt.wikipedia.org/wiki/BIOS>. Acesso em: 22 maio 2018.

WILLIAMS, S. **Free as in freedom**: Richard Stallman's crusade for free. Sebastopol: O'Reilly Media, 2002.

YOKOTE, G. K. L. **O mundo dos nerds**: imagens, consumo e interação. 2014. 2014. 153 f. Dissertação (Mestrado em Antropologia Social) – Faculdade de Filosofia, Letras e Ciências Humanas, Universidade de São Paulo, São Paulo, 2014.

ZÍLIO, C. **Educação pública e opção pelo software livre nas escolas estaduais de Porto Alegre**: um estudo sobre concepções de professores. 2013. 150 f. Dissertação (Mestrado em Educação) – Faculdade de Educação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2013.

ZIMMERMANN, I. **Movimentos sociais e o software livre**. 2007. Dissertação (Mestrado em Educação nas Ciências) – Universidade Regional do Noroeste do Estado do Rio Grande do Sul, Ijuí, 2007.