DOI: 10.12818/P.0304-2340.2025v87p259

MÁQUINAS ABERTAS E RIZOMÁTICAS: REFLEXÕES SOBRE O SOFTWARE LIVRE E DE CÓDIGO ABERTO (FLOSS)

OPEN AND RHIZOMATIC MACHINES: REFLECTIONS ON FREE AND OPEN SOURCE SOFTWARE (FLOSS)

Thiago Oliveira Da Silva Novaes[*]
Otávio Morato De Andrade[**]

RESUMO

Este artigo examina o software livre e de código aberto (Free/Libre and Open Source Software – *FLOSS*) em oposição ao modelo proprietário hegemônico. O *FLOSS* é lido a partir do conceito deleuze-guattariano de *rizoma* e em aproximação com a *máquina aberta* proposta por Simondon. A metodologia utilizada é exploratória, a partir de uma revisão bibliográfica interdisciplinar. Conclui-se que o *FLOSS* promove o livre acesso e a circulação do conhecimento técnico, fomentando a inventividade coletiva e a evolução permanente dos objetos técnicos. O tensionamento da lógica dominante de fechamento/dependência abre espaço para abordagens regulatórias que assegurem a interoperabilidade, a transparência e a soberania tecnológica.

Palavras-chave: Open Source. Software livre. Rizoma. Máquina Aberta. FLOSS

ABSTRACT

This article examines Free/Libre and Open Source Software (FLOSS) in opposition to the hegemonic proprietary model. FLOSS is understood through the deleuze-guattarian concept of the rhizome and in close dialogue with Simondon's notion of the open machine. The methodology is exploratory, based on an interdisciplinary bibliographic review. It concludes that FLOSS promotes free access to and circulation of technical knowledge, fostering collective inventiveness and the evolution of technical objects. The tension against the dominant logic of closure and dependency opens space for regulatory approaches that ensure interoperability, transparency, and technological sovereignty.

Keywords: *Open Source. Free Software. Rhizome. Open Machine. FLOSS*

INTRODUÇÃO

Na era digital, estabeleceu-se como modelo predominante o *software-proprietário ou fechado*: aquele cujo código-fonte é controlado por empresas e inacessível ao público – mediante licenças pagas e restrições de modificação/redistribuição. Ao oferecer soluções variadas às demandas corporativas e/ou pessoais, o software-proprietário difundiu-se mundialmente[1], moldando nosso consumo e relacionamento com as ferramentas tecnológicas. Todavia,

* Doutor em Antropologia Social (UnB, 2016). Professor Colaborador junto ao Mestrado em Avaliação de Políticas Públicas da Universidade Federal do Ceará (UFC).
E-mail: thiago_novaess@yahoo.com.br. ORCID: https://orcid.org/0000-0002-5677-6073.

** Doutorando em Direito na UFMG. Professor da UNIFEMM.
E-mail: otaviomorato@gmail.com. ORCID: https://orcid.org/0000-0002-0541-7353.

1 Exemplos de sistemas pagos e fechados incluem Windows e Pacote Office (Microsoft), macOS e iOS (Apple), Photoshop, InDesign e Premiere Pro (Adobe), AutoCAD (Autodesk), MATLAB (MathWorks), SAP (SAP SE), Oracle Database (Oracle), Salesforce (Salesforce Inc.), e Final Cut Pro (Apple).

estes softwares fechados, de um lado, cerceiam a autonomia de clientes e desenvolvedores, e de outro, excluem usuários que não podem adquiri-los, perpetuando barreiras econômicas. Ademais, a concentração de decisões nas mãos de poucas empresas limita o potencial evolutivo, a inovação e a adaptabilidade técnica dessas ferramentas, promovendo, não raro, a *obsolescência programada* em razão de interesses comerciais[2,3].

Neste cenário, o software livre e de código aberto *(Free/Libre and Open Source Software – FLOSS[4])* aparece como modelo alternativo a partir dos anos 1980/1990, desafiando os limites impostos pelo software-proprietário, ao priorizar o acesso universal, a colaboração e a inovação distribuída. Desenvolvido por comunidades globais de programadores e entusiastas, como a Fundação Apache[5], o projeto GNU/Linux, a comunidade Python[6] e plataformas colaborativas como o GitLab e GitHub[7], o *FLOSS* adota uma lógica descentralizada que permite contribuições e melhorias constantes, promovendo sistemas tecnológicos mais sustentáveis e adaptáveis[8].

O presente artigo tem por objetivo principal explorar aproximações entre o *FLOSS* e os conceitos de *rizoma* e *máquina aberta*, enfatizando a contribuição do software livre e de código aberto para a construção de sistemas tecnológicos mais adaptáveis e colaborativos, que favorecem a integração entre técnica e sociedade. O *rizoma*[9] descreve sistemas abertos e dinâmicos, onde alianças e contribuições coletivas substituem a dependência de uma instância central dominante. Já a *máquina aberta*[10], noção proposta por Gilbert Simondon (1924-1989), destaca a importância de se projetar objetos técnicos modulares e expansíveis. A hipótese é que o FLOSS se insere em ambas as categorias, articulando a

2 O software-proprietário impõe atualizações de sistema que frequentemente tornam versões mais antigas incompatíveis com dispositivos ou aplicativos. Essa prática força os usuários a adquirirem hardware ou software mais recentes para continuar utilizando serviços básicos, mesmo quando os dispositivos ainda estão em pleno funcionamento. Um exemplo bastante atual é o dos celulares que param de receber atualizações, como o iPhone X, que deixou de ser compatível com atualizações do iOS em 2025, limitando sua funcionalidade e segurança.

3 Morato e Novaes, 2024.

4 Embora haja uma sobreposição quase completa entre licenças de software livre e licenças de software de código aberto, há uma forte discordância filosófica entre os defensores dessas duas posições, que será detalhada no capítulo 2 deste trabalho. Sendo tal distinção pouco relevante para fins deste estudo, utilizaremos o termo FLOSS, denominação neutra e genérica, que abarca ambas as categorias.

5 Organização norte-americana sem fins lucrativos que apoia projetos de software de código aberto, como o servidor web Apache HTTP.

6 Linguagem de programação de alto nível amplamente usada e mantida por uma comunidade global de desenvolvedores.

7 Plataformas online para hospedagem de código, colaboração e controle de versões, amplamente utilizadas por desenvolvedores de software.

8 (Stallman, 2021).

9 (Deleuze e Guattari, 2021)

10 (Simondon, 2007)

horizontalidade rizomática das redes de colaboração com a abertura adaptativa seja de elementos, indivíduos ou conjuntos técnicos.

Neste sentido, o trabalho está estruturado em três partes. Inicialmente, é empreendida uma contextualização crítica das práticas de software-proprietário, destacando suas limitações e contradições em relação aos custos, colaboratividade e inovação. Em segundo lugar, o software livre e *open source* é apresentado, mostrando-se como o desenvolvimento cooperativo mitiga os aspectos problemáticos identificados no software-proprietário. A terceira e última parte faz uma leitura do *FLOSS* segundo as chaves analíticas do rizoma e da *máquina aberta*, que permitem refletir sobre novas formas de mentalidade e relação humana com a tecnologia.

A pesquisa tem natureza qualitativa e caráter exploratório, buscando articular um referencial teórico interdisciplinar. O procedimento adotado foi a revisão bibliográfica, com foco na convergência entre filosofia da técnica (Simondon, Deleuze e Guattari) e o ecossistema *FLOSS*. Para a obtenção de novos dados e informações sobre *FLOSS*, foram consultados livros, artigos, repositórios institucionais e bases acadêmicas, como SciELO, HAL e Google Scholar, priorizando publicações revisadas por pares em português e inglês. As buscas abrangeram o período de 2010 a novembro de 2025, utilizando palavras-chave como "open source", "software livre", "licenciamento", "interoperabilidade", "copyleft" e "direito ao reparo". Foram incluídas obras e documentos normativos diretamente relacionados à abertura de código, propriedade intelectual e políticas públicas de software, e excluídas fontes sem autoria identificável. Esse recorte permitiu desenvolver uma análise crítica e atualizada do tema, articulando dimensões técnicas, filosóficas e jurídicas.

Neste sentido, o artigo busca compreender, de maneira interdisciplinar, como o *FLOSS*, enquanto expressão técnica e cultural, pode revelar formas mais abertas e relacionais de interação entre humanos e máquinas. Mais do que uma alternativa técnica, trata-se de um campo de experimentação que reconfigura valores de autonomia, criatividade e compartilhamento do conhecimento, reivindicando deslizamentos jurídicos em temas como licenciamento, soberania digital e direito ao reparo.

1. LIMITAÇÕES DO SOFTWARE-PROPRIETÁRIO

Podemos definir software-proprietário como *um programa de computador licenciado com direitos exclusivos para a empresa desenvolvedora, resultando em uma forte restrição sobre as possibilidades de modificação e adaptação por usuários e terceiros*. Com a popularização de computadores nas décadas de 1970 e 1980, a prática de manter o código-fonte dos softwares inacessível aos usuários tornou-se corrente entre as fabricantes. Assim, passou-se a utilizar patentes, direitos autorais, acordos de licença do usuário final (do inglês: *End User*

License Agreements - EULAs) e outras medidas legais para construir softwares com código fechado, sendo proibida a utilização, cópia, ou redistribuição, em partes ou no todo, sem a permissão do seu detentor[11].

Dentre os principais problemas gerados pelo software-proprietário, destacamos, a seguir: a) a *centralização* das decisões nas mãos das empresas líderes de mercado; b) a imposição de *barreiras econômicas*, dificultando o acesso a tecnologias por usuários e organizações de baixa renda; c) a *limitação da inovação*, causada pelo controle exclusivo do código-fonte pelas empresas desenvolvedoras e d) a prática de *obsolescência programada*, que antecipa deliberadamente a vida útil de um software ou determinada versão.

Problemas do software-proprietário

Problema	Explicação
Centralização	Concentração das decisões na mãos de poucas empresas, comprometendo privacidade, segurança e soberania tecnológica
Barreiras econômicas	Altos custos de licenças excluem usuários e organizações de baixa renda, perpetuando desigualdades sociais e econômicas
Limitação da inovação	O controle exclusivo do código-fonte limita adaptações, engessando a inovação e reduzindo a competitividade
Obsolescência programada	Defasagem no desempenho, atualizações forçadas e fim do suporte forçam a compra de novos produtos/versões

Figura 1: Problemas do software-proprietário. Elaboração própria.

Em primeiro lugar, o monopólio ou oligopólio é recorrente no mercado de software-proprietário, fazendo com que poucas corporações, como Microsoft e Apple, imponham ecossistemas fechados de uso e desenvolvimento. Esse modelo restringe a *interoperabilidade* e gera uma *dependência* das empresas e ferramentas que já lideram o mercado, obrigando usuários e instituições a permanecerem vinculados a elas, com altos custos para migração. Como essas empresas têm exclusividade sobre o código-fonte, elas detêm decisões cruciais sobre aqueles produtos: acréscimo ou eliminação de funcionalidades, integração (ou não) com outros serviços, atualizações, precificação, etc. Esse controle total exercido pelas empresas torna os softwares menos transparentes e auditáveis, acendendo preocupações sobre riscos de segurança e privacidade, sobretudo quando os softwares empregam algoritmos complexos de inteligência

11 Pimentel e Silva, 2014.

artificial[12]. Além disso, a verticalização abala a *soberania tecnológica* de países, deixando governos vulneráveis a sanções comerciais e ingerências em sistemas estratégicos[13].

Um segundo problema decorrente do domínio de softwares-proprietários surge com as *barreiras econômicas* por eles criadas: os altos custos das licenças travam o acesso a sistemas essenciais, especialmente para utilizadores com menos recursos. Escolas públicas, hospitais, organizações não-governamentais (ONGs) e pequenas empresas frequentemente enfrentam dificuldades em adquirir ferramentas tecnológicas, o que perpetua desigualdades sociais e econômicas[14]. Um exemplo paradigmático do impacto econômico do software-proprietário é o sistema *SAP Business One Professional*[15], amplamente utilizado em grandes e médias empresas para gestão integrada de processos. As licenças deste software custam, na época em que se escreve este artigo, cerca de R$500,00 por usuário/mês, valor que pode aumentar substancialmente com a compra de *add-ons* (módulos extras para solucionar demandas específicas) como PCP Avançado[16] (R$ 300 a R$ 800/mês por usuário) WMS Avançado[17] (custo de R$ 200 a R$ 500/mês por usuário) e SPED Fiscal e Contábil[18] (R$1.200/mês por empresa; usuários ilimitados). Tudo isso desconsiderando os custos de implementação inicial e treinamento de funcionários que, não raro, ultrapassam centenas de milhares de reais[19].

Considerando o salário mínimo atual do Brasil (R$ 1518,00 em 2025) e que cerca de 90% dos brasileiros ganha uma renda inferior a R$ 3.422,00 por mês[20], não seria exagero afirmarmos que, *em muitos casos, o custo de algumas licenças de software-proprietário supera até mesmo o salário do funcionário utilizador daquela licença*.

Enquanto isso, a SAP SE[21], proprietária do software e atualmente a maior empresa europeia, tem valor de mercado estimado em R$ 1,41 trilhões, figurando recorrentemente em rankings das maiores e mais lucrativas do mundo. Com

Alves e Morato, 2022.

13 Rashid, 2023.

14 Sullivan, 2010.

15 O SAP Business One é um software de planejamento de recursos empresariais (do inglês: Enterprise Resource Planning - ERP) desenvolvido pela SAP SE, voltado para pequenas e médias empresas, que integra funções como finanças, vendas, logística e operações em uma única plataforma.

16 Planejamento e Controle de Produção (PCP) avançado é um add-on para controle de ordens de fabricação e planejamento de materiais, ideal para indústrias.

17 Warehouse Management System (WMS) avançado é um add-on para rastrear estoques e aperfeiçoar a gestão logística e de armazéns.

18 Automação das obrigações fiscais e contábeis brasileiras, como ICMS, IPI e ECD.

19 ALFA Sistemas de Gestão, 2024.

20 Equipe Toro Investimentos, 2024.

21 A alemã SAP SE, fundada em 1972 por ex-funcionários da IBM, é uma das líderes globais em

monopólio de mercado e imenso poder econômico, corporações como a SAP SE reforçam a dependência de empresas menores e até governos em relação a soluções proprietárias[22,23]. Tal dinâmica evidencia as barreiras econômicas levantadas pelo software-proprietário, que contribui para perpetuar desigualdades tecnológicas e privilegia organizações bilionárias em detrimento da democratização do acesso à tecnologia.

Em terceiro lugar, situamos a *limitação da inovação*. O controle exclusivo do código-fonte pelas empresas desenvolvedoras limita drasticamente a capacidade de inovação. Ao impedir que usuários aprimorem seus softwares, as grandes desenvolvedoras podem atrasar o progresso tecnológico, especialmente em áreas que exigem soluções personalizadas, como educação e saúde. Além de cercear o potencial criativo de desenvolvedores independentes, o imenso poder das líderes de mercado também dificulta a concorrência, seja encarecendo o custo de entrada de novos atores na arena comercial, seja absorvendo *start-ups* ou concorrentes ascendentes, que trazem ideias e conceitos renovadores[24].

Por último, e não menos importante, o software-proprietário facilita a obsolescência programada (OP): *um conjunto de práticas adotadas por fabricantes que intentam deliberadamente diminuir a vida útil (ou a percepção dela) de um produto, incentivando os consumidores a substituírem seus bens por novos mais frequentemente do que seria tecnicamente necessário*. No caso dos softwares, a OP impele os usuários a adquirir novas versões de produtos ou substituir os equipamentos que rodam esses programas (servidores, laptops, smartphones, etc). Atualizações incompatíveis com sistemas mais antigos e o fim do suporte técnico são estratégias amplamente utilizadas para acelerar o ciclo de consumo, gerando custos desnecessários e, eventualmente, contribuindo para o aumento de resíduos eletrônicos. Essa prática penaliza especialmente usuários e empresas de menor porte, que não têm condições de arcar com custos recorrentes de atualizações ou novos equipamentos[25].

Desta maneira, o software-proprietário pode ser compreendido como um modelo de desenvolvimento tecnológico que privilegia interesses corporativos, reduzindo a acessibilidade, a inovação e autonomia do usuário e do desenvolvedor. Saleh mostrou que, embora existam soluções em software livre para grande parte das necessidades empresariais, com vantagens em termos de funcionalidades e custos, elas são comparativamente pouco utilizadas em relação ao software proprietário[26]. Isso se deve, principalmente, à *realimentação positiva*: os

software corporativo, oferecendo soluções com foco em inteligência artificial, computação em nuvem e integração de processos empresariais em mais de 130 países.

22 Olivieri, 2024.

23 Protska, 2023.

24 OSI, 2025.

25 Morato e Novaes, 2024

26 Saleh, 2004

softwares dominantes formam uma rede de usuários/empresas da qual seus utilizadores são incapazes de se desligar. O campo da economia comportamental ajuda a explicar melhor este movimento: seja pelo *viés de conformidade social*, no qual os usuários seguem a maioria para evitar conflitos ou sensação de isolamento social, ou pelo *viés do status quo*, que favorece a manutenção de soluções já adotadas, evitando custos psicológicos de mudança[27]. Tais vieses, aliados ao *efeito de rede* – que aumenta o valor de um produto à medida que mais pessoas o utilizam[28] – consolidam o domínio do software proprietário, mesmo quando alternativas livres são econômica, estratégica e socialmente mais vantajosas.

Verifica-se que modelos centralizadores que inibem o acesso e a inovação opõem-se às crescentes demandas de uma sociedade em rápida transformação, que busca se relacionar de maneira mais ética e sustentável com a tecnologia. É preciso, como alertou Gilbert Simondon, uma tomada de consciência acerca dos objetos técnicos, compreendendo que seu desenvolvimento deve estar alinhado a uma cultura que valorize a *abertura* e a *adaptabilidade* da tecnologia, permitindo sua evolução em sintonia com as necessidades humanas e ambientais, explorando suas *margens de indeterminação*, não seus *automatismos*[29]. Desta perspectiva, assumir as limitações do software-proprietário é um passo essencial para repensar nossa cultura técnica e buscar alternativas que favoreçam um progresso participativo, sustentável e democrático.

2. COLABORAÇÃO E INOVAÇÃO NO *OPEN SOURCE* E SOFTWARE LIVRE

O compartilhamento de saberes técnicos tem raízes históricas profundas e foi gradualmente ampliado por estruturas colaborativas e eventos globais. Os séculos XVII e XVIII assistiram à disseminação das sociedades e revistas científicas[30]. Já no século XIX, deu-se o advento do sistema internacional de exibições industriais, inaugurado pela Grande Exposição de 1851 em Londres. Desde as patentes cruzadas[31] na indústria automobilística norte-americana no início do século XX até iniciativas como a ARPANET[32] nos primórdios da computação, a

27 Morato, 2019.

28 Stobierski, 2020.

29 Simondon, 2007, p. 33.

30 Kronick, 1976

31 O licenciamento cruzado de patentes é um acordo entre duas ou mais empresas para permitir o uso mútuo de suas patentes. Em vez de negociar royalties, as partes envolvidas concordam em compartilhar suas inovações, permitindo que cada empresa utilize as tecnologias protegidas por patentes da outra. Este tipo de licenciamento é particularmente comum em indústrias onde a tecnologia é altamente interdependente, como a eletrônica, telecomunicações e biotecnologia.

32 A ARPANET (Advanced Research Projects Agency Network) foi a primeira rede de computadores a usar o protocolo de comutação de pacotes, criada pelo Departamento de Defesa dos Estados Unidos em 1969. Inicialmente desenvolvida para conectar instituições de pesquisa, foi

lógica colaborativa impulsionou avanços tecnológicos fundamentais. No campo da informática, podemos destacar, como precursores do código aberto, a cultura *Do it Yourself* (em português: Faça Você Mesmo) e a ideologia *Hacker* dos anos 1960/1970.

Antes de mais nada, é necessário traçar uma diferença entre os movimentos *software livre* e *open source*, esclarecendo como estes termos serão usados. O primeiro, criado pelo ativista Richard Stallman em 1983 e institucionalizado em 1985 por meio da *Free Software Foundation (FSF)*, enfatiza a *liberdade total* do usuário de "executar, acessar, modificar e redistribuir cópias com ou sem modificações", com foco em princípios éticos e filosóficos[33]. Já o *open source* (em português: código aberto), liderado pela *Open Source Iniciative (OSI)*, fundada em 1998, privilegia a disponibilização pública do código-fonte para colaboração e modificação, focando nos benefícios práticos do desenvolvimento[34].

Assim, *todo software livre é considerado open source, mas nem todo open source é considerado livre*, já que algumas licenças, ainda que em código aberto, impõem o uso exclusivo em determinadas plataformas ou proíbem redistribuições modificadas. Para fins deste estudo, tal distinção é pouco relevante, já que ambos os movimentos convergem para o uso e o desenvolvimento de softwares mais abertos, acessíveis e colaborativos. Optamos, portanto, por adotar o termo *FLOSS (Free/Libre and Open Source Software)*, nomenclatura mais neutra em relação às divergências entre OSI e FSF e que é capaz de abarcar, com completude, ambos os projetos.

Existem ainda outras iniciativas que pregam a gratuidade do software, mas que não se confundem com o *FLOSS*. A nomenclatura *Freeware* refere-se a programas gratuitos que, todavia, *não* oferecem acesso ao código-fonte (podemos citar, como exemplos, o Skype e o Adobe Acrobat Reader). Por sua vez, o *Public Domain Software* é aquele software sem direitos autorais, mas que não necessariamente assegura as liberdades centrais de alteração de código fonte (ex.: SQLite e 4DOS). Tais iniciativas, embora importantes em outros contextos, não serão abordadas neste estudo, cujo foco recai exclusivamente sobre o software livre e de código aberto.

Desenvolvidos, respectivamente, nas décadas de 1980 e 1990, os conceitos de software livre e *open source (FLOSS)* são uma resposta ao modelo de software-proprietário praticado por empresas como IBM, Microsoft e Apple, que já comercializavam softwares e hardwares desde as origens da computação comercial. Inicialmente, a troca de conhecimento sobre *FLOSS* era feita a partir

o precursor direto da internet moderna. A ARPANET permitiu a troca de informações entre computadores de maneira descentralizada, marcando um avanço significativo na comunicação digital.

33 Sullivan, 2010, tradução nossa.

34 OSI, 2025.

de fóruns online, listas de e-mail e repositórios FTP[35], que permitiam o compartilhamento de arquivos e projetos de código. Na década de 2000, emergiram novas plataformas e comunidades, dentre as quais destacamos Launchpad (fundada em 2004), Reddit (2005) e GitHub (2008), possibilitando uma colaboração global e descentralizada de desenvolvedores – que passaram a corrigir falhas e evoluir os sistemas sem uma autoridade central. Exemplos incluem o sistema operacional Linux, o servidor Apache e o gerenciador de contêineres Kubernetes[36], amplamente utilizados em ambientes corporativos, acadêmicos e pessoais. Portanto, a expansão do *FLOSS* foi impulsionada pela internet, que viabilizou comunidades online onde se trocavam códigos e ideias sobre desenvolvimento.

A relevância do *FLOSS* reside em sua capacidade de atender às demandas de uma sociedade tecnológica em constante transformação. O modelo oferece flexibilidade, redução de custos e adaptabilidade, características fundamentais para enfrentar desafios contemporâneos como bugs, falhas de segurança e interoperabilidade[37] com novas ferramentas. Além disso, ao promover um ambiente mais aberto, o software livre e de código aberto democratiza o acesso a tecnologias essenciais, permitindo que indivíduos e organizações contribuam para o avanço técnico de forma colaborativa[38].

Características do software livre e/ou open-source

Característica	Vantagem
Descentralização	Promove transparência, segurança e confiança ao distribuir decisões e responsabilidades
Redução de custos / Inclusão	Acessibiliza as tecnologias para indivíduos e organizações de diferentes contextos econômicos
Inovação colaborativa	Impulsiona avanços contínuos ao reunir contribuições de diversas perspectivas em um esforço coletivo
Longevidade	Possibilidade de manutenção, reparo e atualização contínuos, evitando a obsolescência

Figura 2: Características do software livre. Elaboração própria.

35 Repositórios FTP são sistemas que utilizam o protocolo de transferência de arquivos, tecnologia que permite a movimentação de dados entre servidores e dispositivos de forma estruturada e segura.

36 Desenvolvido originalmente pelo Google e agora mantido pela Cloud Native Computing Foundation, Kubernetes é um sistema de gerenciamento de ambientes virtuais de código aberto.

37 Ordenamentos jurídicos como o *Digital Markets Act (DMA)*, impuseram obrigações de interoperabilidade a plataformas dominantes (União Europeia, 2022).

38 Kon, 2001.

Neste sentido, o *FLOSS* pode ser lido como um fato jurídico relevante, capaz de reconfigurar categorias clássicas do Direito da Tecnologia, da Propriedade Intelectual e também de áreas conexas, como o Direito Administrativo, o Direito Econômico, o Direito da Concorrência e o Direito Constitucional. Em um cenário global de busca por soberania digital e transparência, alguns países têm estimulado o uso do *FLOSS*, reconhecendo suas vantagens econômicas e estratégicas. Desde 2005, a França migrou a Gendarmerie Nationale, órgão militar com relevância estratégica, para o Ubuntu Linux, reduzindo custos em 40% e incentivando o uso de ferramentas como OpenOffice e Firefox. Em 2019, a União Europeia lançou o projeto EU-FOSSA 2, investindo 2,6 milhões de euros para aumentar a segurança de *FLOSS,* por meio de eventos, auditorias e programas de recompensa para encontrar e corrigir bugs.

Em 2024, a Suíça, por meio da EMBAG (Lei federal sobre o uso de meios eletrônicos para o cumprimento de tarefas governamentais), tornou obrigatório o uso de software *open source* em projetos públicos. No mesmo ano, a Alemanha revisou suas leis de acesso online para aprimorar a interoperabilidade e reduzir a dependência de software proprietário, estimulando, por exemplo, a colaboração através do OpenCoDE, um repositório central para compartilhamento de código entre governos[39]. Em 2025, a Dinamarca iniciou um projeto-piloto com o LibreOffice em seu sistema nacional de gestão eletrônica de processos, e o estado alemão de Eslésvico-Holsácia decidiu substituir as plataformas Microsoft Office e Microsoft Teams por soluções de código aberto[40].

Nos anos 2000 e no início da década de 2010, o Brasil ocupou posição de destaque nesse debate, implementando um amplo conjunto de estímulos ao software livre: criação do sede do Fórum Internacional de Software Livre (FISL); desenvolvimento do e-Gov[41], do Portal do Software Público Brasileiro[42] e de vários sistemas abertos pelo SERPRO[43]; anúncio de preferência por softwares livres na Esplanada[44], dentre outras medidas. Essa agenda, voltada à democratização tecnológica, à autonomia informacional e à mitigação do *vendor lock-in*[45], perdeu força a partir de 2016, com a reorientação de contratações públi-

39 Vaughan-Nichols, 2024.

40 Faustino, 2025.

41 Programa federal iniciado em 2000 para estruturar serviços públicos digitais e estabelecer diretrizes de tecnologia para a administração pública.

42 Plataforma criada em 2007 para disponibilizar, compartilhar e manter softwares livres desenvolvidos pelo e para o governo federal.

43 Serviço Federal de Processamento de Dados, empresa pública federal responsável por desenvolver e prover soluções tecnológicas para a administração pública brasileira.

44 Em 2003, a Casa Civil orientou todos os ministérios a priorizarem software livre nas soluções de TI do governo federal.

45 Condição de dependência tecnológica estruturada por soluções proprietárias, que limita a portabilidade e restringe a autonomia diante de um fornecedor específico.

cas para fornecedores proprietários[46]. Em 2018, após dezoito edições, o FISL entrou em hiato. Atualmente, embora o Governo Federal siga recomendando a adoção de soluções abertas, essas medidas não dispõem da relevância político-institucional que marcou a fase anterior[47]

3. O OPEN SOURCE: ESQUEMA RIZOMÁTICO E MÁQUINA ABERTA

3.1 Open source/software livre como rizoma

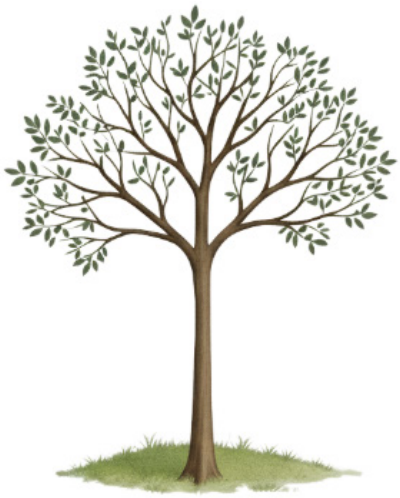
Viu-se, anteriormente, que os projetos de software livre e *open source* promovem a colaboração ativa entre usuários e desenvolvedores, tornando o código e a estrutura dos objetos acessíveis para modificações coletivas e melhorias contínuas. A partir dessa perspectiva, gostaríamos de sugerir que *o FLOSS reflete a lógica rizomática descrita por Deleuze e Guattari*. Em *Mil Platôs*, publicado em 1980, Gilles Deleuze e Félix Guattari opuseram a figura da *árvore* ao *rizoma*, destacando que, enquanto a primeira representa um modelo hierárquico e centralizado, o segundo simboliza uma rede dinâmica e interconectada. Vejamos como essa lógica rizomática é aplicável ao FLOSS.

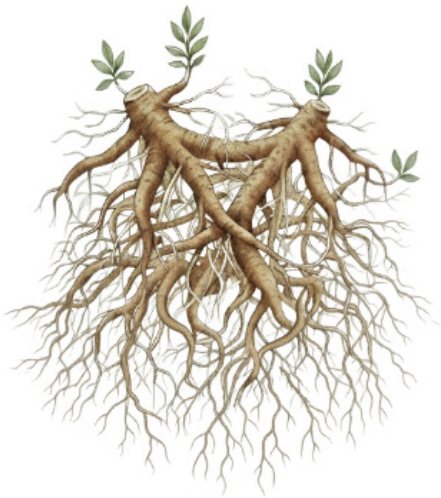
O rizoma é definido por Deleuze e Guattari como uma forma de pensamento e organização que contrasta com a estrutura hierárquica e linear tradicional, adotando uma multiplicidade de trajetórias possíveis, onde cada ponto pode se conectar a qualquer outro. Esta lógica opõe-se à tradição cartesiana de pensamento verticalizado, linear e binário representada pela *árvore*, que organiza suas raízes e ramificações sobre uma base fixa e centralizada. No rizoma, não há pontos fixos, árvores ou raízes; tudo é composto de *linhas* que se conectam e se expandem. O rizoma, neste sentido, é uma estrutura aberta, descentralizada e interconectada, que rejeita hierarquias rígidas e fixas em favor das múltiplas possibilidades de conexão. Ao contrário da lógica arbórea, que se baseia em ramificações hierarquizadas, o rizoma não possui um "ponto de partida" único nem tampouco um "destino" fixo: ele se desenvolve de maneira horizontal e em todas as direções. Tal multiplicidade permite que diferentes elementos coexistam e se relacionem de forma dinâmica, criando novas configurações e trajetórias inusitadas, ao invés de seguir uma ordem previsível ou predefinida[48].

46 Lisboa e Marilene, 2019.

47 Brasil, 2025.

48 Deleuze e Guattari, 2021, p. 30.





Figuras 3 e 4: Arvore e Rizoma: Comparação entre dois modelos estruturais. A árvore representa uma estrutura centralizada, hierárquica e direcionada, enquanto o rizoma exemplifica conexões horizontais, múltiplas trajetórias e ausência de hierarquia. Elaboração própria.

Mas como essa lógica rizomática se manifesta no software livre e de código aberto? Analisaremos cada um dos seis princípios propostos pelos autores, que descrevem as seguintes "características aproximativas" do rizoma: (a) *conexão*, que descreve como o rizoma estabelece relações entre diferentes elementos, criando redes dinâmicas que não dependem de hierarquia fixa; (b) *heterogeneidade*, que reforça que essas conexões ocorrem entre elementos diversos e incompatíveis; c) *multiplicidade*, segundo o qual o rizoma é um sistema de intensidades em constante crescimento e transformação, mas sem unidade central; (d) *ruptura a-ssignificante* que realça a reorganização em novos pontos, adaptando-se e criando novas trajetórias e interações, ainda quando interrompido; (e) *cartografia*, que compara o rizoma a um *mapa* sempre em construção, que pode ser modificado e acessado por múltiplos pontos de entrada e saída e, por fim, (f) *decalcomania*, que rejeita cópias ou modelos fixos, enfatizando a transformação contínua e a adaptação às circunstâncias[49]

Passemos à análise do *FLOSS* à luz dos aspectos rizomáticos elencados. Primeiro, a *conexão* é evidente na forma como o código aberto permite que desenvolvedores de diferentes contextos colaborem, unindo ideias e habilidades em plataformas globais como GitHub. Essa conectividade dá-se de maneira *heterogênea*, não se limitando a um grupo específico: programadores experientes, iniciantes, curiosos, empresas, governos e comunidades contribuem democraticamente, alimentando o movimento por meio de diversas perspectivas e intensidades. Portanto, o código aberto é múltiplo, recebendo colaborações diversas

49 Deleuze e Guattari, 2021, p. 30.

que não se organizam em torno de uma unidade central: antes, operam como um sistema expansivo e descentralizado, no qual cada contribuição amplia suas possibilidades.

Além disso, o *FLOSS* incorpora a *ruptura a-ssignificante*, pois, mesmo quando um projeto é abandonado ou interrompido, ele pode ser retomado por outros, que reorganizam suas linhas de desenvolvimento de maneiras inovadoras. Um exemplo é o *LibreOffice*, lançado em 2010 como uma ramificação do projeto original *OpenOffice.org* (lançado em 2000). Em 2009, quando a Oracle comprou a Sun Microsystems (e junto dela os direitos do OpenOffice.org), alguns desenvolvedores e colaboradores do OpenOffice.org deixaram o projeto controlado pela Oracle para montar o LibreOffice, criando um *fork* (ramificação) do projeto original[50].

O *FLOSS* também se expressa rizomaticamente pelo princípio da *cartografia*, funcionando como um *mapa aberto*, com múltiplas entradas e possibilidades de modificação e adaptação conforme as necessidades do utilizador. O *mapa*, segundo Deleuze e Guattari, é uma representação dinâmica e aberta da realidade, que privilegia conexões e multiplicidades em vez de fixar pontos ou estabelecer hierarquias, permitindo constantes transformações e reconfigurações:

> O mapa é aberto, é conectável em todas as suas dimensões, desmontável, reversível, suscetível de receber modificações constantemente. Ele pode ser rasgado, revertido, adaptar-se a montagens de qualquer natureza, ser preparado por um indivíduo, um grupo, uma formação social[51].

Ou seja, dentro dessa *rede viva* que é o rizoma, o *mapa* age como um guia: ele não copia a realidade como uma foto; mas cria caminhos novos e se adapta conforme é utilizado. Cada projeto de software livre, como o GNU/Linux ou o Apache, funciona como *mapa*, em constante construção e expansão, onde os participantes exploram novas possibilidades e criam soluções inovadoras sem a imposição de um centro controlador. Ao contrário do *mapa*, o software proprietário opera como um *decalque*: algo rígido, com começo e fim, que reproduz estruturas limitadoras de acesso e inovação.

Finalmente, o *FLOSS* rejeita o modelo fixo ou imutável, alinhando-se ao princípio da *decalcomania*: não reproduz e nem copia estruturas, mas promove uma uma experimentação ancorada no real, que se traduz na transformação contínua dos projetos de softwares, refletindo a fluidez e a adaptabilidade características do rizoma.

Portanto, verifica-se que o *FLOSS* materializa todos os seis princípios: a maneira como seus projetos são desenvolvidos independe de uma autorida-

50 Vignoli, 2023.

51 Deleuze e Guattari, 2021, p. 30.

de central, permitindo conexões múltiplas e dinâmicas entre seus elementos. O *FLOSS* emerge como uma prática que encarna multiplicidade, criatividade e liberdade, subvertendo os paradigmas de controle e exclusão que dominam o cenário do software-proprietário, como mostrado no primeiro capítulo deste trabalho.

O famoso sistema operacional GNU/Linux ilustra como o *FLOSS* personifica a totalidade dos aspectos rizomáticos. Primeiramente, ele é *conectável* em qualquer ponto: sua estrutura permite que módulos, funcionalidades ou distribuições sejam interligados de maneira livre, sem hierarquias fixas. Ele também é *heterogêneo*, pois conecta desenvolvedores, empresas, estudantes e usuários finais, integrando contextos diversos em um ecossistema global. Sua multiplicidade manifesta-se nas inúmeras distribuições (como Ubuntu, Fedora e Debian), cada uma com propósitos específicos, mas interligadas por um núcleo comum. O GNU/Linux é *a-ssignificante*, pois não subordina-se a uma estrutura centralizada ou única; qualquer usuário pode adaptá-lo às suas necessidades sem depender ou subordinar-se a um "General"[52]. Além disso, ele é ruptível: novas ramificações sempre podem surgir, como *forks*[53] que criam soluções independentes, mantendo a vitalidade do sistema. Por fim, ele é *cartográfico*, permitindo que cada desenvolvedor crie um *mapa* único de soluções com base em seu próprio ambiente e necessidades, sem seguir um modelo fixo. O GNU/Linux, portanto, materializa a ideia de rizoma Deleuziana-Guattariana:

> O rizoma é uma antigenealogia. É uma memória curta ou uma antimemória. O rizoma procede por variação, expansão, conquista, captura, picada. Oposto ao grafismo, ao desenho ou à fotografia, oposto aos decalques, o rizoma se refere a um mapa que deve ser produzido, construído, sempre desmontável, conectável, reversível, modificável, com múltiplas entradas e saídas, com suas linhas de fuga (...) Contra os sistemas centrados (e mesmo policentrados), de comunicação hierárquica e ligações preestabelecidas, o rizoma é um sistema acentrado não hierárquico e não significante, sem General, sem memória organizadora ou autômato central, unicamente definido por uma circulação de estados[54]

Por outro lado, a estrutura rígida e hierárquica da *árvore* compara-se ao modelo de software proprietário, que limita a flexibilidade e a multiplicidade, promovendo uma relação de subordinação a uma autoridade central.

Ainda aproveitando a terminologia de Deleuze-Guattari, é cabível sustentar que o *FLOSS* pode ser lido como um grande *agenciamento*, no qual diferentes elementos (desenvolvedores, usuários, ferramentas, repositórios, *etc*)

52 Deleuze, Guattari, 2021, p 43

53 *Forks* são ramificações de um projeto de software onde o código original é copiado e desenvolvido de forma independente, permitindo inovações ou adaptações específicas. No Linux, forks como o CentOS Stream ou AlmaLinux ilustram essa flexibilidade do *FLOSS* (Pettle, 2024).

54 Deleuze e Guattari, 2021, p. 44

se conectam e interagem para produzir softwares adaptáveis e evolutivos. Esse processo rompe os estratos das barreiras econômicas e técnicas, *desterritorializando* tecnologias computacionais clássicas para *reterritorizá-las* em repositórios e fóruns que privilegiam a inventividade e a colaboração. Ao permitir modificações e contribuições coletivas, o software livre e de código aberto ainda exemplifica uma *máquina desejante*, movida pelo desejo de criar, compartilhar e transformar.

A capacidade de incorporar mudanças, aspecto tipicamente rizomático, também é notável no *FLOSS*, cuja adaptabilidade e receber aportes colaborativos permite sua evolução contínua. Cada contribuição ao código, seja uma correção de bug ou a criação de uma nova funcionalidade, funciona como uma *linha de fuga*, uma força criativa que rompe com estruturas fixas, permitindo a emergência de novas possibilidades e transformações. É a linha de fuga que expande as possibilidades do projeto e permite sua adaptação contínua. Pensando nestes termos, a estrutura rizomática do *FLOSS* estimula reapropriações e imprevisibilidades, desafiando a rigidez dos modelos proprietários.

Portanto, ao adotar conexões horizontais e colaborativas, o *FLOSS* tende a mitigar os problemas diagnosticados no modelo proprietário – marcado por centralização, custos elevados e inovação calculada. O software livre e de código aberto promove transparência e evolução contínua, enquanto o software proprietário frequentemente impõe obsolescência programada, restringindo atualizações e gerando dependência econômica e tecnológica. Ademais, ao eliminar barreiras de acesso, o *FLOSS* subverte a exclusão típica do software fechado, democratizando a tecnologia e fomentando a evolução, orientada por necessidades genuínas da comunidade e não por interesses exclusivamente comerciais.

3.2 Máquina aberta

Em 2017, o filósofo francês Gilbert Simondon foi homenageado pela edição anual dos Encontros Mundiais de Software Livre. O evento, que teve lugar na cidade de Saint-Étienne, terra natal de Simondon, reuniu especialistas para discutir novas formas de relação com a tecnologia a partir da colaboração e inventividade – temas que inspiraram Simondon, notadamente no livro *Do modo de existência dos objetos técnicos (MEOT)*, publicado em 1958[55]. Nesta obra, Simondon[56]oferece uma base filosófica para compreender a técnica, almejando a reintrodução do ser técnico na cultura, em que é "preciso definir o objeto técnico em si mesmo, através do processo de concretização e sobredeterminação funcional que lhe dá sua consistência ao fim de sua evolução, provando que não poderia ser considerado um mero utensílio".

55 Novaes, 2017

56 Simondon, 2007, p. 37

Com efeito, verifica-se uma convergência entre o funcionamento do software livre e de código aberto e as ideias de Simondon. Neste sentido, gostaríamos de dar ênfase à possibilidade de ler o *FLOSS* como uma *máquina aberta*, termo utilizado por Simondon para se referir a objetos técnicos dotados de alta tecnicidade[57], adaptáveis ao meio e que evoluem em resposta a contingências. Não por acaso, máquina aberta é, para o filósofo, a expressão de uma mentalidade técnica, uma certa forma de pensar e se relacionar com os objetos técnicos, que reconhece sua evolução e integração como parte da cultura humana:

> Se procurarmos o signo da perfeição da mentalidade técnica, podemos reunir em um único critério a manifestação dos esquemas cognitivos, das modalidades afetivas e das normas de ação: o da abertura; a realidade técnica é eminentemente suscetível de ser continuada, completada, aperfeiçoada, prolongada[58].

Neste sentido, a máquina aberta é um sistema ajustável, com uma *margem de indeterminação* que potencializa a interação com seres técnicos e humanos, calibrando e amplificando sua funcionalidade de acordo com novas demandas. A máquina aberta é dotada de uma estrutura expansível, permitindo que seu operador intervenha em sua estrutura – por meio do acréscimo ou retirada de componentes a depender das necessidades[59]. A atualização permanente da máquina aberta evita sua obsolescência, permite usos renovados (como o acréscimo de funcionalidades), como um organismo em crescimento, e contribui para um modelo diferenciado de progresso técnico – que não reproduz a lógica linear da produção industrial, mas favorece uma evolução aberta e relacional da técnica.

Cotejando as características apresentadas no Capítulo 2 com o pensamento de Simondon, pode-se sustentar o enquadramento do *FLOSS* como "máquina aberta". Verifica-se que o modelo de código aberto é altamente expansível e sensível ao *meio associado*, colocando a máquina em constante transformação. Com efeito, vimos que o código aberto permite uma participação ativa e criativa dos usuários, que decidem *quando* intervir e *quais* funcionalidades aprimorar, a partir das dificuldades prévias (por exemplo, *bugs*) ou modificações do ambiente (novas necessidades para o uso pessoal/comercial, ou demandas de interoperabilidade com outros softwares).

Além disso, o fato de o sistema ser aberto a modificações e intervenções humanas reflete outra grande aposta da filosofia de Simondon: o papel do ser humano como organizador e *mediador* desses sistemas, atuando como uma espécie de "maestro" que harmoniza e integra as interações entre os objetos técnicos e entre os próprios usuários: "Está *entre* as máquinas que operam

57 Simondon, 2007, p. 33

58 Simondon, 2022, p. 33, grifo do autor.

59 Simondon, 2007

com ele"[60]. Para que esse organizador-inventor da "orquestra técnica" cumpra plenamente sua função, é essencial que ele desenvolva e exercite uma *cultura técnica*, que reconheça e valorize os objetos técnicos para além da sua mera funcionalidade[61]

No texto *Mentalidade técnica*[62], Simondon mostrou que a máquina aberta beneficia-se de uma estrutura *reticular*. Enquanto externamente o objeto técnico está conectado a uma rede mais ampla de sistemas técnicos e sociais, sua estrutura interior também é interconectada, com partes interdependentes e relativamente independentes em suas funções. No *FLOSS*, cada componente pode ser modificado, adaptado ou substituído por outros usuários sem comprometer o sistema como um todo. Assim como os "nós" de uma rede, cada módulo desempenha uma função específica e *bugs* podem ser corrigidos de maneira colaborativa, preservando a integridade geral do sistema. Esse acesso a elementos, indivíduos e conjuntos técnicos permite uma lógica reticular que assegura continuidade e adaptabilidade do ser técnico, ajudando a prolongar a sua vida útil e a torná-lo mais eficiente, e mesmo plurifuncional.

Por outro lado, podemos tomar os softwares proprietários como exemplos de máquinas fechadas – objetos técnicos caracterizados pelo isolamento, rigidez e resistência à intervenção humana[63]. O funcionamento das máquinas fechadas tende ao automatismo completo, sem permitir modificações ou adaptações pelos usuários. Como mostra Simondon na *Entrevista sobre a tecnologia com Yves Deforge:*

> Ao fechamento material das soldas, dos rebites e dos lacres de garantia, soma-se um fechamento mais essencial e mais alienante: o objeto já não é decodificável, nem compreensível como resultado de uma operação de construção. Já não se pode ler nele a operação construtora. Ele é estranho como uma língua estrangeira. Compreende-se, nessas condições, por que um objeto assim pode ser tratado como um escravo mecânico. Não se busca compreender a linguagem do escravo, mas apenas obter dele um serviço específico. Sobre o objeto técnico em situação de alienação, o painel de controle e os órgãos de comando bastam para a operação prática de uso no contexto de um trabalho determinado[64].

Ainda que seus exemplos pertençam ao campo mecânico, a crítica de Simondon ao fechamento como forma de alienação técnica pode ser estendida ao universo computacional, onde o bloqueio de acesso ao código repete esse mes-

60 Simondon, 2007, p. 34, itálico no original.

61 Simondon, 2007, p. 32.

62 O texto *Mentalité Technique* foi trazido à público por Jean-Hugues Barthélémy e Vincent Bontems em 2006 na *Revue philosophique*, ganhando tradução para o português em 2022, no livro Máquina Aberta, organizado por Novaes, Smarieri e Vilalta.. Estima-se que as reflexões deste texto foram registradas por Simondon nas décadas de 1960-1970.

63 Simondon, 2017

64 Simondon, 2014, pp. 65-66, tradução nossa

mo mecanismo. Com efeito, observa-se que a estrutura dos softwares-proprietários é de difícil modificação, mesmo para utilizadores experientes, pois estes não possuem acesso ao código fonte. Trata-se, portanto, de *máquinas fechadas* que restringem a interação criativa do usuário com o sistema, impedindo que este evolua de maneira colaborativa e se reintegre continuamente ao meio técnico e social – resultando no fenômeno da *obsolescência*.

É precisamente contra essa lógica da *máquina fechada* que surgem as propostas legislativas contemporâneas de Direito ao Reparo (Right to Repair). Tais iniciativas visam garantir legalmente o acesso a manuais, peças e, crucialmente, ao *código-fonte* e ferramentas de diagnóstico, alinhando-se diretamente ao ideal de uma tecnologia mais aberta, durável e passível de intervenção humana[65,66]

Para além das vantagens apontadas no uso do FLOSS, chamamos atenção para uma *experiência* de envolvimento direto do usuário com a modificação e personalização do objeto técnico. Não se trata apenas de atender às suas necessidades práticas, mas também de despertar um *senso de realização criativa*, um *desfrute tecnoestético*[67]. O código aberto transforma o ato técnico de *programar um software* em uma experiência rica, no qual funcionalidade e beleza se entrelaçam, não para contemplação, mas para a ação, reafirmando a dimensão humana e inventiva do *fazer tecnológico*. A lógica do código aberto permite que o usuário seja mais do que um utilizador, associando o processo de criação ao de preservação e crescimento do ser técnico, prevendo o ajuste, a ampliação e a reinvenção do objeto. Trata-se de um novo modo de se vivenciar a experiência técnica, no qual a *relação* com o objeto engendra uma comunicação com o objeto, não de apropriação, resultando em um regime de coindividuação.

Tal conceito contribui para uma reaproximação efetiva entre cultura e tecnologia, atualmente apartadas nas sociedades informatizadas. Sobretudo com o advento das TICs, o desenvolvimento exponencial dos objetos digitais voltados para o consumo alienado e baseados em automatismos, fez com que estas avançassem independentemente da cultura, e tal desconhecimento perpetuou uma relação escravocrata com a técnica. Segundo Simondon[68], a cultura estabeleceu-se como um "sistema de defesa contra as técnicas", por meio de oposições maniqueístas entre teoria e prática, contemplação e ação, beleza e utilidade, ou mesmo cultura e civilização:

65 [...] se o objeto estiver aberto, isto é, se o gesto do usuário, por um lado, pode ser um gesto inteligente e bem adaptado, conhecendo as estruturas internas, se por outro lado o reparador [...] puder perpetuamente manter novas as peças que se desgastam, então não há datamento, não há envelhecimento. Sobre uma base que é uma base de durabilidade ou pelo menos de grande solidez, podemos instalar peças que terão de ser substituídas, mas que, em todo o caso, deixam intacto o esquema fundamental e que até permitem melhorá-lo [...] (Simondon, 2014, p. 401-402, tradução nossa)

66 Morato, 2025.

67 Simondon, 1998.

68 Simondon, 2007, p. 31.

> A cultura não entende a máquina; é inadequado à realidade técnica porque considera a máquina como um bloco fechado e o funcionamento mecânico como um estereótipo iterativo. [...] A cultura é injusta com a máquina, não apenas nos seus julgamentos ou nos seus preconceitos, mas no próprio nível do conhecimento: a intenção cognitiva da cultura em relação à máquina é substancializante; a máquina está presa nesta visão redutiva que a considera completa em si mesma e perfeita, o que a faz coincidir com o seu estado atual, com as suas determinações materiais[69]

Em MEOT, Simondon mostra que, no processo de concretização, os objetos técnicos evoluem analogamente aos seres vivos, aproximando-se (sem, contudo, se equipararem completamente) de um sistema natural. O *FLOSS* alinha-se a essa ideia de organicidade crescente: o software é evoluído e concretizado por meio do gesto humano, distinguindo-se objeto técnico primitivo, que era rígido e isolado. Além disso, muito antes do advento da internet, Simondon[70] já sugeria a necessidade de disponibilização de "blocos de conhecimento técnico", ou seja, conhecimentos compartilhados e aperfeiçoados comunitariamente, com foco na abertura e adaptabilidade dos objetos técnicos.

Simondon nos desafia a respeitar e entender a tecnicidade dos objetos, invertendo a ética relacional: em vez de "como devemos nos comportar com os objetos técnicos?", ele propõe um dever moral de conhecê-los. Assim, a ética torna-se uma avaliação das relações que estabelecemos com as máquinas e, por meio delas, com o mundo[71]. Essa perspectiva relacional se configura como um meio de libertação da *alienação*, cujas raízes estariam na ignorância sobre a "natureza" e a "essência" das máquinas[72]. Desenvolver uma relação profunda com os objetos técnicos, compreendendo seu funcionamento e propósito, abriria caminhos para uma regulação mais consciente do acoplamento entre técnica e sociedade humana:

> A consciência dos modos de existência dos objetos técnicos deve ser alcançada por meio do pensamento filosófico, que deve cumprir um dever por meio deste trabalho análogo ao que cumpriu para a abolição da escravidão e a afirmação do valor da pessoa humana[73].

O software livre se apresenta como um objeto técnico mais evoluído tecnicamente que o software proprietário justamente por corrigir, em sua essência técnica, uma dupla concepção equivocada da cultura técnica alienada. A primeira que concebe a máquina mais evoluída como sendo a dotada de mais automatismo; ao contrário, conforme demonstra o software livre, sua natureza aberta permite sua atualização permanente e adaptação a novas realidades de

69 Simondon, 2017, p. 201.
70 Simondon, 2017, p. 110-111.
71 Alombert, 2023.
72 Simondon, 2017, p. 18.
73 Simondon, 2017, p. 9.

hardware, culminando em novas funcionalidades. Em segundo lugar, convida o humano a se relacionar com a técnica não como um senhor que dá ordens a escravos, mas sim aquele que estabelece uma comunicação com as máquinas, como um maestro dos conjuntos técnicos.

CONSIDERAÇÕES FINAIS

Este artigo examinou, inicialmente, as limitações do software proprietário, atentando para os aspectos provocados pela lógica fechada dominante, como a *centralização*, *exclusão econômica*, *restrição à inovação* e *obsolescência programada*. Mostrou-se que o modelo proprietário consolida monopólios, dificulta o acesso a ferramentas tecnológicas por indivíduos e organizações de baixa renda e limita a evolução técnica ao restringir modificações e adaptações do código-fonte. Em contraste, o software livre e de código aberto *(FLOSS)*, analisado no segundo capítulo, demonstrou ser uma alternativa sustentável e colaborativa, promovendo acesso universal, inovação contínua e participação coletiva por meio do código aberto. Ao priorizar a transparência e a flexibilidade, o *FLOSS* mitiga as barreiras econômicas e técnicas, além de desafiar os padrões excludentes impostos pelo modelo proprietário, ensejando um ecossistema tecnológico mais democrático e inclusivo.

A análise do software livre e de código aberto sob a perspectiva de Deleuze e Guattari identificou a presença dos seis princípios rizomáticos descritos pelos autores: *conexão, heterogeneidade, multiplicidade, ruptura a-significante, cartografia e decalcomania*. Tais princípios revelam uma estrutura descentralizada, adaptável e em constante evolução, na qual desenvolvedores e usuários contribuem coletivamente para o aprimoramento e transformação do código. A capacidade do FLOSS de incorporar múltiplas perspectivas, de se reorganizar mesmo após interrupções e de operar como um *mapa* vivo e aberto reflete a lógica rizomática, recusando hierarquias centralizadas e articulando a emergência de soluções tecnológicas colaborativas, espontâneas e inovadoras. Nesse contexto, o *FLOSS* opera como um sistema dinâmico que amplia as possibilidades de interação e (re)invenção tecnológica.

O conceito simondoniano de *máquina aberta* oferece uma visão complementar, tomando o *FLOSS* como um objeto técnico *expansível, modular* e *adaptável*: trata-se de uma máquina cuja *essência técnica* permite ajustes e prolongamentos contínuos por meio da exploração de sua margem de indeterminação, ou seja, o acesso ao seu código que permite uma interação criativa entre desenvolvedores (ao contrário dos softwares proprietários, que remetem às máquinas fechadas por restringir intervenções e favorecer a obsolescência). O *FLOSS* valoriza nossa relação com a tecnologia por meio da inventividade, alinhando-se à aposta de Simondon em uma relação não escravocrata com a máquina, sugerindo o humano enquanto "regente" da técnica.

À luz da articulação dos teóricos mobilizados, pode-se afirmar que *o FLOSS representa um estágio técnico mais avançado em relação ao software proprietário*. Seu caráter autocorretivo e colaborativo prolonga a vida útil dos objetos técnicos, reduz a obsolescência e favorece uma cultura tecnológica orientada pela transparência, pela partilha do conhecimento e pela integração entre técnica e cultura. É nítido o potencial do *FLOSS* para subverter os limites do modelo fechado, possibilitando novas formas de criação, uso e revalorização dos objetos técnicos.

Neste sentido, o *FLOSS* não deve ser lido apenas como solução técnica: trata-se de um *projeto ético e político*, que ajuda a repensar a relação entre humanos, tecnologia e o meio ambiente. Conectando múltiplos agentes e promovendo intervenções criativas, o *FLOSS* tende a materializar uma relação sociotécnica mais horizontal e dinâmica, apontando para uma relação mais consciente e equilibrada com o mundo técnico.

No campo jurídico, essa estrutura colaborativa deve materializar-se, por exemplo, nos modelos de licença. De um lado, licenças *copyleft* (como a *GNU General Public License - GPL*) asseguram que o caráter *rizomático* se perpetue nas obras derivadas, criando um bem comum digital protegido. De outro, licenças permissivas (como *MIT* e *Apache*) focam na máxima modularidade da *máquina aberta*, permitindo que o código seja reincorporado livremente, inclusive em software proprietário, fomentando a interoperabilidade e a inovação comercial.

Para além do licenciamento, o *FLOSS* oferece um fundamento robusto para o debate jurídico sobre o Direito ao Reparo (Right to Repair) e o combate à obsolescência, baseada no fechamento legal que controla o ciclo de vida do produto (via EULAs, propriedade intelectual e leis antipirataria)[74]. Neste sentido, o tensionamento da lógica proprietária dominante demanda novos modelos jurídicos para valorização do uso, da manutenção e da soberania tecnológica, alinhado a uma economia mais circular e sustentável.

Concordando a necessidade de mudança "na mirada filosófica sobre o objeto técnico"[75], o presente estudo tem caráter eminentemente conceitual e não contempla avaliação empírica sistemática. Uma agenda de pesquisa futura poderia avançar em análises mais aplicadas, investigando, por exemplo: (a) o impacto concreto da adoção de *FLOSS* no setor público brasileiro (especialmente em compras, interoperabilidade e soberania digital); (b) uma análise dogmática aprofundada dos efeitos jurídicos de licenças *copyleft* vs. permissivas em contratos; e (c) a relação entre o acesso ao código e as propostas de auditoria algorítmica e direito ao reparo. Assim disposta, essa agenda teria muito a contribuir sobre as decisões políticas de definição de licenciamento de software público,

74 Morato, 2025.

75 Simondon, 2007, p. 38.

bem como orientar a compra e o desenvolvimento de software levando-se em conta a alta tecnicidade da máquina aberta, não os interesses mercadológicos incrustados no software proprietário.

REFERÊNCIAS

Alfa Sistemas de Gestão. Preço do SAP Business One: Licenças, Implementação, Suporte e Add-ons. ALFA Sistemas de Gestão. 12/03/2024. Disponível em: https://www.alfaerp.com.br/blog/preco-do-sap-business-one-licencas-implementacao-suporte-e-add-ons/. Acesso em 12/11/2025.

Alves, Marco Antônio Sousa e Morato, Otávio. Da 'caixa-preta' à 'caixa de vidro': o uso da explainable artificial intelligence (XAI) para reduzir a opacidade e enfrentar o enviesamento em modelos algorítmicos. **Direito Público**, vol. 18, no. 100, 2022, pp. 1-21. Disponível em: https://www.portaldeperiodicos.idp.edu.br/direitopublico/article/view/5973. DOI: https://doi.org/10.11117/rdp.v18i100.5973. Acesso em 12/11/2025.

Brasil. Ministério da Gestão e da Inovação em Serviços Públicos. Boas práticas para minimizar aprisionamento (lock-in) em nuvem. 10/6/2025. Disponível em: https://www.gov.br/governodigital/pt-br/infraestrutura-nacional-de-dados/ambiente-tecnologico/nuvem/materiais-de-apoio/boas-praticas-para-minimizar-aprisionamento-em-nuvem. Acesso em 16/11/2025.

Deleuze, Gilles; Félix Guattari. **Mil Platôs:** Capitalismo e Esquizofrenia vol 1. São Paulo: Editora 34, 2021.

Equipe Toro Investimentos. Pirâmide Salarial: qual a média salarial e quem é considerado rico no Brasil? **Blog da Toro Investimentos.** 28/11/2024. Disponível em: https://www.toroinvestimentos.com.br/blog/piramide-salarial. Acesso em 12/11/2025.

Faustino, Felipe. Mais um país: Alemanha abandona Microsoft por softwares de código aberto. **Tecnoblog,** 16/06/2025. Disponível em: https://tecnoblog.net/noticias/mais-um-pais-alemanha-abandona-microsoft-por-softwares-de-codigo-aberto/. Acesso em 12/11/2025.

Kon, Fabio. O Software Aberto e a Questão Social. Relatório Técnico RT-MAC-2001-07, Departamento de Ciência da Computação, Instituto de Matemática e Estatística, Universidade de São Paulo, 2001. Disponível em: https://www.ime.usp.br/~kon/papers/RT-SoftwareAberto.pdf. Acesso em 12/11/2025.

Kronick, David. **A History of Scientific and Technical Periodicals:** The Origins and Development of the Scientific and Technical Press, 1665-1790. 2nd ed., Scarecrow Press, 1976.

Lisboa, Flávio; Beatriz, Marilene. Um recorte da história do software livre no Estado brasileiro: um estudo de caso do Programa de Software Livre do SERPRO. **Tear: Revista de Educação Ciência e Tecnologia,** vol. 8, no. 2, 2019.

Disponível em: https://periodicos.ifrs.edu.br/index.php/tear/article/view/3476. DOI: https://doi.org/10.35819/tear.v8.n2.a3476. Acesso em 13/11/2025.

Morato, Otávio. NudgeRio: Um caso de aplicação de ciência comportamental às políticas públicas. **Cadernos do Desenvolvimento Fluminense** n. 16, 2019, pp. 111-124. Disponível em: https://www.e-publicacoes.uerj.br/cdf/article/view/52711. DOI: https://doi.org/10.12957/cdf.2019.52711. Acesso em 12/11/2025.

Morato, Otávio; Novaes, Thiago. Serialização de Componentes e Obsolescência Programada: diálogos com Simondon pelo Direito ao Reparo. *In:* Thiago Novaes. (Org.). **Simondon 100 años: Pensamiento Transductivo.** 1ªed.Bogotá: Editorial Aula de Humanidades, Utedé, 2024 , p. 137-160.)

Morato, Otávio. What is design for repair and how does it help combat obsolescence in the context of the circular economy? **Revista de la Facultad de Derecho de México**, vol. 75, no. 292, pp. 201-230. Disponível em: https://www.revistas.unam.mx/index.php/rfdm/article/view/90064. DOI: https://doi.org/10.22201/fder.24488933e.2025.292.90064. Acesso em 13/11/2025.

Novaes, Thiago. Encontro Mundial de Software Livre – Homenagem a Simondon. **Site Gilbert Simondon.** 7 de julho de 2017. Disponível em: https://gilbert-simondon.wordpress.com/2017/07/07/encontro-mundial-de-software-livre-homenagem-a-simondon/. Acesso em 11/11/2025.

Olivieri, Fernando. SAP vira a empresa de tecnologia mais valiosa da Europa. **Exame**, 23/10/2024. Disponível em: https://exame.com/tecnologia/sap-vira-a-empresa-de-tecnologia-mais-valiosa-da-europa/. Acesso em 10/11/2025.

OSI (Open Source Initiative). The Open Source Definition. **Open Source Initiative**. 16/02/2024. Disponível em: www.opensource.org/osd. Acesso em 10/11/2025.

Pettle, Amy. AlmaLinux Promises Continued RHEL Compatibility. **Linux Magazine**, Issue 278, 2024. Disponível em: www.linux-magazine.com/Issues/2024/278/AlmaLinux. Acesso em 10/11/2025.

Pimentel, Luiz Otávio; Figueiredo e Silva, Cláudio Eduardo Regis de. Conceito Jurídico de Software, Padrão Proprietário e Livre: Políticas Públicas. **Seqüência: Estudos Jurídicos e Políticos,** vol. 35, no. 68, 2014, pp. 291-329. Disponível em: https://periodicos.ufsc.br/index.php/sequencia/article/view/2177-7055.2013v35n68p291. DOI: https://doi.org/10.5007/2177-7055.2013v35n68p291. Acesso em 10/11/2025.

Protska, Olga. As 10 Empresas de TI Mais Lucrativas do Mundo - 2025. **FXSSI Blog**, 14/04/2023. Disponível em: https://fxssi.com/as-10-empresas-de-ti-mais-lucrativas-do-mundo-2025. Acesso em 10/11/2025.

Saleh, Amir Mostafa. Adoção de tecnologia: um estudo sobre o uso de software livre nas empresas. Dissertação de mestrado, Universidade de São Paulo, Facul-

dade de Economia, Administração e Contabilidade, Departamento de Administração, 2004. Disponível em: https://teses.usp.br/teses/disponiveis/12/12139/tde-06122004-123821/pt-br.php. Acesso em 10/11/2025.

Simondon, Gilbert. **Sur la technique (1953-1983).** Paris: Presses Universitaires de France, 2014.

Simondon, Gilbert. **On the mode of existence of technical objects.** Trad. Cécile Malaspina e John Rogove. Univocal Publishing, 2017.

Simondon, Gilbert. **El modo de existencia de los objetos técnicos**. Trad. Margarita Martínez e Pablo Rodríguez. Buenos Aires: Prometeo Libros, 2007.

Simondon, Gilbert. A mentalidade técnica. Trad. Thiago Novaes e Evandro Smarieri. *In* **Máquina Aberta:** a mentalidade técnica de Gilbert Simondon [org. Thiago Novaes, Lucas Paolo Vilalta e Evandro Smarieri]. Dialética: São Paulo, 2022.

Simondon, Gilbert. Sobre a tecno-estética: Carta a Jacques Derrida. *In* **Tecnociência e Cultura –** ensaios sobre o tempo presente. São Paulo: Estação Liberdade, 1998, pp. 253-266.

Stallman, Richard. FLOSS and FOSS. GNU Operating System. **Free Software Foundation**, 2021. Disponível em: https://www.gnu.org/philosophy/floss-and-foss.html. Acesso em 10/11/2025.

Stobierski, Tim. What Are Network Effects? **Harvard Business School Online**, 12/11/2020. Disponível em: https://online.hbs.edu/blog/post/what-are-network-effects. Acesso em 10/11/2025.

Sullivan, John. Free Software Is a Matter of Liberty, Not Price. **Free Software Foundation**. 16/03/2010. Disponível em: www.fsf.org/free-software-is-a-matter-of-liberty. Acesso em 9/11/2025.

União Europeia. Regulamento (UE) 2022/1925 do Parlamento Europeu e do Conselho, de 14/09/2022. Jornal Oficial da União Europeia, L 265, 12 out. 2022, pp. 1-66. Disponível em: data.europa.eu/eli/reg/2022/1925/oj. Acesso em 10/11/2025.

Vignoli, Italo. A Brief History of LibreOffice: the origin story of the Open Source Office solution that ensures you always have access to your data and control over your creativity. **Opensource.com**, Red Hat, 7/2/2023. Disponível em: https://opensource.com/article/23/2/history-libreoffice. Acesso em 10/11/2025.

Vaughan-Nichols, Steven. Switzerland federal government requires releasing its software as open source. **ZDNet**. 23/07/2024. Disponível em: www.zdnet.com/article/switzerland-federal-government-requires-releasing-its-software-as-open-source/. Acesso em 10/11/2025.

Recebido em: 06/01/2025

Aprovado em: 17/11/2025