Aluno: Pedro G. Branquinho
Orientador: Dr. Wei-Liang Qian

Softwares Livres na Academia e na Indústria

Lorena, São Paulo

22 de junho de 2021

Resumo

Demonstrou-se como é possível construir uma série de aplicações baseada em softwares de licença livre, à partir de um sistema aberto, o Linux com inteface EXWM - Emacs X Window Manager. Além disso, foi propiciado casos reais de aplicações na Indústria e no investimento privado, autônomo. Bem como, utilizações na Academia, à nível de lecionar, e pequisa. Sustenta-se que a economia aberta possui similaridade estrutural ao movimento Open Source e seu desenvolvimento, o que aponta que essa é e continuará a ser, paulatinamente mais, o paradigma de desenvolvimento econômico tecnológico. Assim, imprescindível à formação do engenheiro.

Palavras-chaves: software livre. automação. freqtrade. idústria. academia.

Lista de ilustrações

Sumário

1 Introdução

Na formação de um engenheiro físico, o qual, por definição, é um profissional generalista, os softwares abertos (FOSS - Free and Open Source Software) e a participação da comunidade Open Source são detrimentais para sua formação.

A diversidade os quais softwares extensíveis acarretam (subseção 2.1.1) podem mudar completamente a experiência do usuário, e o trazer mais próximo do papel de desenvolvedor. Essa experiência não necessita de ser exclusiva de cientistas da computação ou profissionais de TI. Pois, a programação pode ser encarada tanto como ciência e arte (KNUTH, 1968).

Os Softwares Abertos possuem quatro liberdades pétreas seção 2.1, garantindo os direitos de estudo, cópia, modificação e redistribuição.

Bem como a ciência se beneficia com seus rápidos avanços, de uma comunidade global de participantes, com as mais distintas especializações profissionais. Também, beneficia-se a computação com a comunidade aberta, e especialização eclética, tanto de membros quanto de softwares.

1.1 Objetivo

Demonstramos a defassagem que um profissional de engenharia apresentaria, sem forte formação dentro da computação. Ademais, ao partir da gama de aplicações, em estado da arte, as quais são partilhadas de forma aberta e livre, pretende-se reinterar o caso da necessidade de compreensão do fenômeno dos softwares abertos. Pois, essa lógica e dinâmica não possui paralelos nem na economia, nem na comunidade científica (HIPPEL; KROGH, 2003; PETERS, 2009).

Há até debates acirrados sobre o sentido de Open Science, um termo que recentemente se popularizou e o qual constitui claro paralelo com o movimento Open Source. Porém, mal compreendido e, raramente, debatido sob esse prisma, dentro das ciências sociais. Cita-se o mais citado dos artigos na busca no Google Scholar “The future(s) of open science”, o qual somente cita três vezes o termo Open Source, em tom dismissivo (MIROWSKI, 2018). Assim, argumenta-se que a formação básica do engenheiro na área da computação precisa ser sólida, bem como o entendimento das forças que moldaram esse movimento, de forma a poder entender o futuro em que caminhamos, de forma crítica.

1.2 As interconexões das aplicações e o OS

1.2.1 Por quê o GNU/Linux nos importa?

Discutiremos introdutoriamente na monografia, o que é o sistema operacional GNU/Linux, e o por quê de ser a porta de entrada à comunidade Open Source. Primeiramente, o GNU/Linux é o primeiro e maior sucesso da lógica de negócio Open Source (TU et al., 2000; WEST; DEDRICK, 2001) - dessa forma, utilizá-lo é uma maneira de entender como a lógica de negócio dependente de comunidades abertas funcionam (FINK, 2003).

Além do mais, o novo usuário-desenvolvedor de GNU/Linux, inerentemente, tem de aprender sobre outros softwares os quais vêm conjunto à sua distribuição - o que aumenta sua adoção (WEST; DEDRICK, 2001). Desta forma, sente-se compelido a participar e estender os comportamentos do programa (HERTEL; NIEDNER; HERRMANN, 2003).

Para aquém da sua inicial expressão filosófica de liberdade, o GNU/Linux hoje é o standard em empresas te alta tecnologia. E, sua lógica de negócio, por mais que desafie os moldes empresariais econômicos do status quo, sem sombra de dúvidas funciona, traz resultados às empresas, a economia e à comunidade. Usar-se do GNU/Linux, então, é uma maneira de se inserir nesse novo contexto lógico econômico (MOODY, 2009; HIPPEL; KROGH, 2003; PETERS, 2009).

1.2.2 Como as aplicações de alto nível se beneficiam de um *OS*

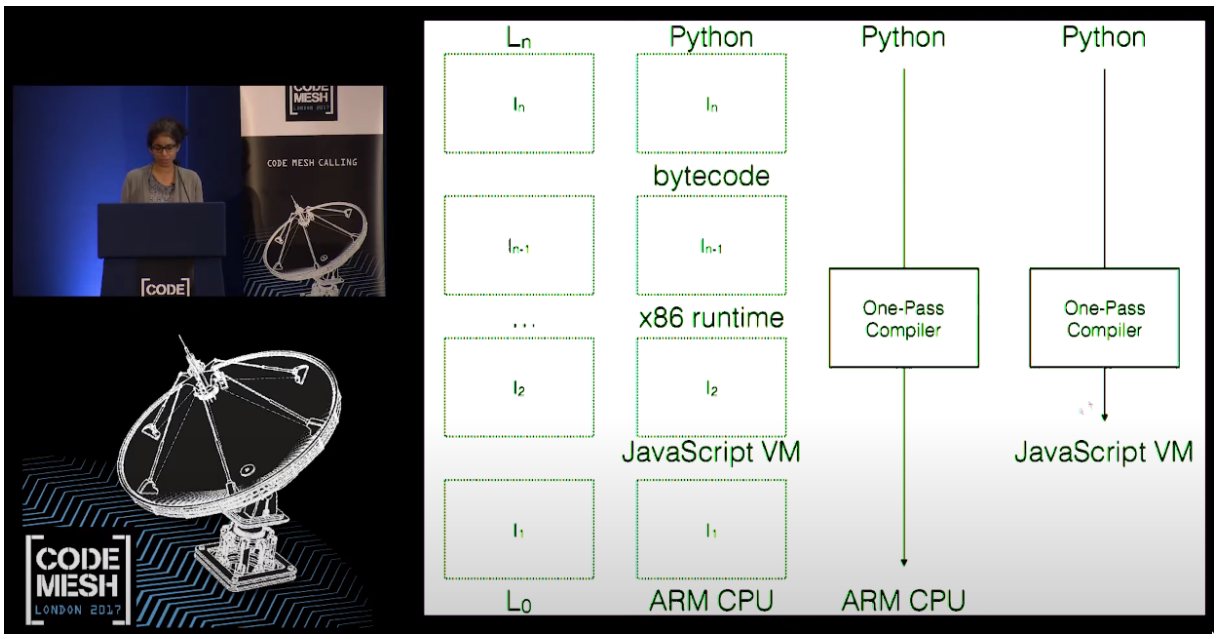
Na hierarquia de softwares e aplicações, os sistemas operacionais podem ser vistos como um meta-aplicação.

> "The evaluator, which determines the meaning of expressions in a programming language, is just another program." (ABELSON; SUSSMAN, 1996)

Existem níveis, ou camadas, de abstração em virtualmente qualquer aplicação. Ou seja, o conceito de meta-programação e torres de interpretadores é um problema um tanto quanto comum, e possui implicações diretas no uso dos softwares.

Os sistemas operacionais, como o GNU/Linux, uma camada essencial nessa torre de interpretadores. Em especial, eles se comunicam com firmwares - softwares de baixa expressividade e alta performance, os quais controlam hardwares. Também, comunicam-se com softwares de alta expressividade, os quais comprazem as aplicações escritas, ou estendidas, pelo usuário-desenvolvedor do sistema. Por conseguinte, o Sistema Operacional possui um papel de mediador entre softwares de alto e baixo nível - como são chamados pela literatura - o que configura-os como uma espécie de middleware.

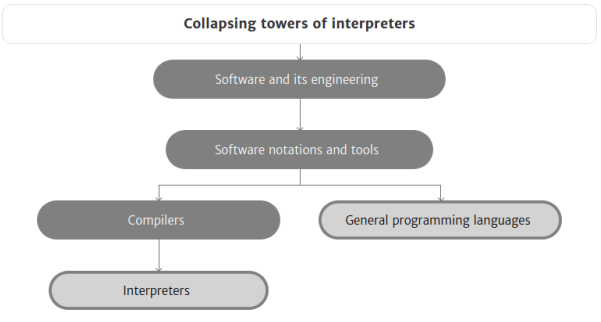
Figura 1 – Esquemática de uma torre de interpretadores



Code Mesh, apresentação Towers of Interpreters, por Nada Amin

O problema característico de concatenar sistemas de softwares um sob o outro introduz complexidades em termos de manter compatibilidade entre versões de programas e sua performance. O estudo desses comportamentos e suas soluções teóricas possuem um ramo próprio, desvinculado, por exemplo, de quais linguagens compõe a torre, ou qual aplicação estamos lidando (AMIN; ROMPF, 2017). O que se interessa é com o comportamento final do sistema, e se é possível colapsar o sistema.

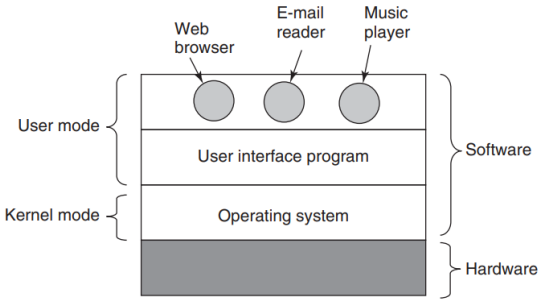
Figura 2 – Categorização do estudo de torres de interpretadores



Fonte: (AMIN; ROMPF, 2017)

Finalmente, os O.S. consistem em grandes colapsadores de torres de interpretação. Encontram-se encarregados de colapsar firmware, middlewares e softwares de alto nível. Por fim, o quão bem um sistema operacional conduz essa tarefa, tanto mais a experiência do usuário final é facilitada.

Figura 3 – Torre de interpretadores e os Sistemas Operacionais.



Fonte: (TANENBAUM; BOS, 2015)

> If every application programmer had to understand how all these things work in detail, no code would ever get written. Furthermore, managing all these components and using them optimally is an exceedingly challenging job. For this reason, computers are equipped with a layer of software called the operating system, whose job is to provide user programs with a better, simpler, cleaner, model of the computer and to handle managing all the resources just mentioned. (TANENBAUM; BOS, 2015)

1.3 Da influência da educação na adoção

É claro, na literatura, de que a adoção dos softwares open source no setor industrial dependem da competência, e do quão profundo seus conhecimentos precisam ser para resolução de seus problemas (LI; TAN; YANG, 2013; GALLEGO et al., 2015; SPINELLIS; GIANNIKAS, 2012).

Ao mesmo tempo, a adoção depende diretamente das inclinações intrínsecas da equipe de TI (RACERO; BUENO; GALLEGO, 2021).

Por fim, por mais que cursos profissionalizantes no tópico aumente a intensidade e rapidez de sua adesão, por alunos (RACERO; BUENO; GALLEGO, 2020) e profissionais inclinados à autonomia (GALLEGO et al., 2015), a adoção final está mais ligada a fatores intrínsecos de motivação. No entanto, é importante notar a existência de um efeito-rede em adoções (SPINELLIS; GIANNIKAS, 2012) e.g., quanto mais semelhantes adotam OSS, maior a chance de adoção.

1.4 Performance e futuro do open source como motivos de adoção

Delineia-se que existe um ganho em performance ao se utilizar a plataforma GNU/Linux em comparação ao Windows (SULAIMAN; RAFFI, 2021). Porém, mais importante ainda, é o fato de que os principais benefícios não estão no sistema opera-

cional em si, mas no treino que se obtém quando se utiliza um sistema totalmente dependente de ferramentas e comunidades abertas.

Pois, é nesse momento que se dá a transformação profissional indivíduo, ao mesmo tempo que se aumenta a chance das empresas que esse profissional irá trabalhar de adotarem OSS. Mesmo que, as empresas em geral, e as de software em particular, beneficiam-se grandemente da comunidade aberta e livre ao mesmo tempo que não precisam e não incentivam nenhum direto incentivo ao trabalho da comunidade (HAUGE; SØRENSEN; CONRADI, 2008).

1.5 O perfil do engenheiro e dos usuários de FOSS

É notável que dado a profundidade que se procura a dar a formação, ainda em nível de graduação, do engenheiro físico, espera-se que ele seja um ideal candidato ao uso de FOSS. Pois, necessita de ambos de inclínio a autonomia (SCHRAPE, 2019; RACERO; BUENO; GALLEGO, 2020), quanto profundidade em seu trabalho técnico, para fomentar a necessidade e adoção de FOSS, em nível individual (LI; TAN; YANG, 2013; GALLEGO et al., 2015).

1.6 Como podemos alavancar o potencial de OSS na Indústria e Academia

No presente trabalho, utilizou-se de diversas demonstrações de conceitos, onde o autor desenvolveu e estendeu aplicações desenvolvidas num contexto livre e aberto. Também, discute-se como se integrar e acrescentar à comunidade aberta. Nota-se o quão fundamental e significante são as conexões profissionais que se faz nesse nicho.

2 Revisão Bibliográfica

2.1 Open Source

Qualquer programa que permita o usuário-programador ter as seguintes liberdades:

1. Direito de rodar o programa, como você desejar, para qualquer fim.
2. Direito ao acesso ao código-fonte, para estudá-lo.
3. Direito de cópia e distribuição.
4. Direito à modificação do software.

De maneira prática, a comunidade Open Source, fundamentalmente, se baseia no compartilhamento de suas configurações. As vantagens de existirem inúmeras outras pessoas utilizando o mesmo software é de que a melhoria da fronteira do programa é expandida de forma acrescida, em comparação a de um time restrito de usuários.

2.1.1 Diversidade

Dado que um direito fundamental dos softwares livres é a modificação e propagação das versões modificadas, existe uma diversidade de expressividade, sem paralelos em outras áreas da tecnologia.

Por exemplo, uma parte de software fundamental na configuração de um computador é seu gerenciador de interfaces (Window Manager). Onde, um programa é devotado a gerenciar como outros programas gráficos devem se dispor na tela de computador.

Enquanto sistemas operacionais (Operational Systems) privados, como Windows e MacOS possuem versões lançadas frequentemente - vinte e cinco versões lançadas de Windows. O Windows possui apenas quatro versões, com suporte ativo (MICROSOFT…, 2021).

São vinte lançamentos de MacOS, e quatro verões mantidas (MACOS…, 2021).

Essa estreiteza de versões se dá, dentre os fatores, pois os usuários são cerciados do direito de extender ou alterar os comportamentos programados no sistema. Assim, vítimas do suporte descontinuado e de sua atualização de versões restritivas.

Em contra partida, existem, paralelamente, por volta de 278 distribuições de

Linux (LINUX..., 2021). Onde, existem as distribuições raízes, com princípios e filosofias de desenvolvimentos teóricos e práticos diferentes.

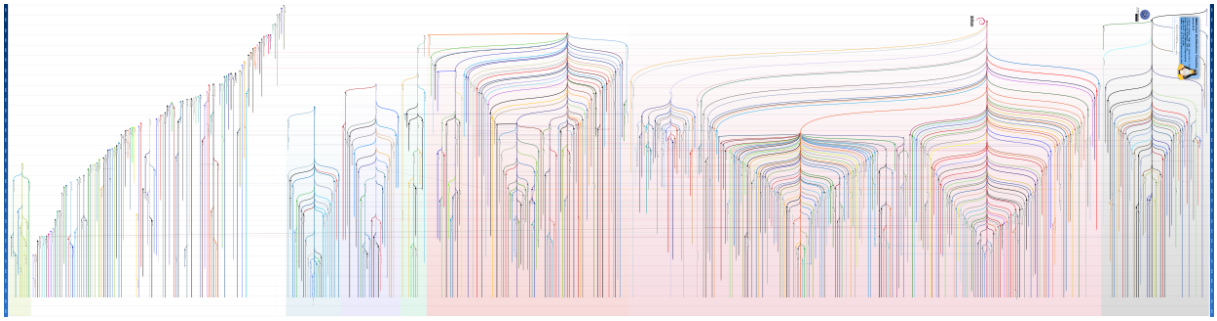
Assim, bem como em qualquer outro escopo de software, a variação dos softwares abertos e livres (FOSS) sempre serão superiores aos monopolizados.

2.2 O Linux

Existem distribuições raízes de linux, das quais muitas distribuições existem como ramificações. Nomeia-se, de forma genêrica, devido aos princípios base de uma classe de distribuições, como famílias. Cita-se algumas das mais influentes e populares, Red Hat Linux, Debian, CentOS, Fedora,Pacman-based, OpenSUSE, Gentoo-based, Ubuntu-based, Slackware, Open Sourced-based e as distribuições Independetes.

É possível apreciarmos visualmente a riqueza de distribuições pela Figura 4.

Figura 4 – Genealogia Distribuições Linux



Histórico de evolução das distribuições Linux (LINUX..., 2021)

2.2.1 Origem Histórica

O projeto do GNU/Linux iniciou-se separadamente, por duas frentes. O GNU - abreviação de, GNU' s Not Unix - por usuários revoltados com o sistema de segurança dos computadores do MIT (Laboratory of Computer Science - LCS) (STALLMAN, 2002; EMACS..., 2021). Dentre eles, o ainda ativo Richard Stallman, após já dez anos de evolução do editor de texto (EMACS..., 2021).

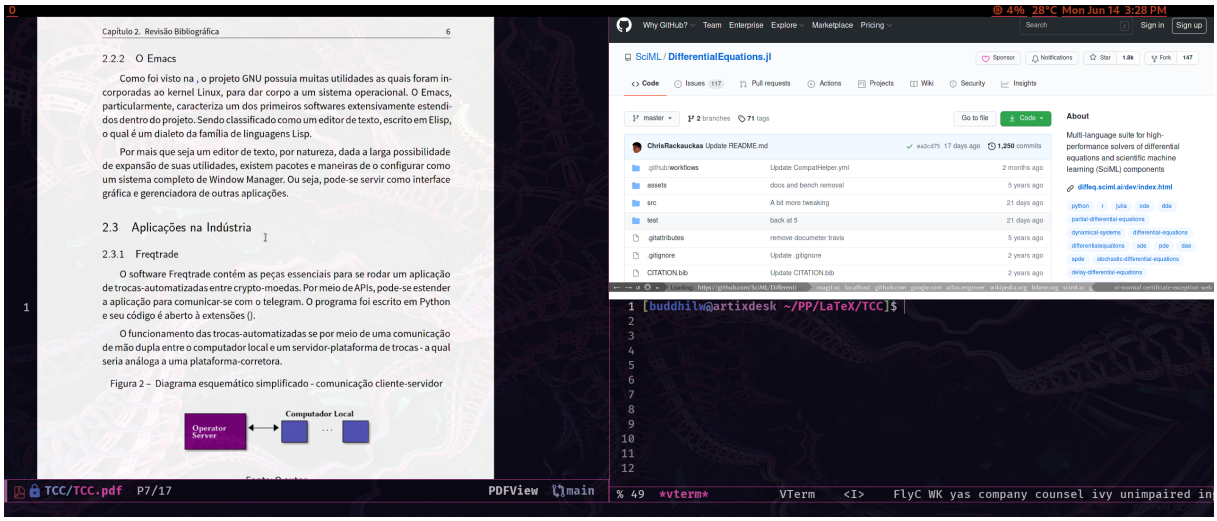
Paralelamente, Linus Torvalds desenvolveu um sistema operacional portável aberto, como sua tese de mestrado (TORVALDS, 1997). Por fim, houve uma junção dos projetos, os quais colaboravam o Linux, como sistema operacional, e GNU com todas as aplicações utilitarias do sistema (STALLMAN, 1997).

2.2.2 O Emacs

Como foi visto na , o projeto GNU possuia muitas utilidades as quais foram incorporadas ao kernel Linux, para dar corpo a um sistema operacional. O Emacs, particularmente, caracteriza um dos primeiros softwares extensivamente estendidos dentro do projeto. Sendo classificado como um editor de texto, escrito em Elisp, o qual é um dialeto da família de linguagens Lisp.

Por mais que seja um editor de texto, por natureza, dada a larga possibilidade de expansão de suas utilidades, existem pacotes e maneiras de o configurar como um sistema completo de Window Manager. Ou seja, pode-se servir como interface gráfica e gerenciadora de outras aplicações.

Figura 5 – EXWM - Emacs X Window Manager



Fonte: foto do ambiente de WM do autor.

A Figura 5 é um exemplo do ambiente de desktop totalmente manuseado por meio do Emacs, utilizando-se do EXWM. Pode-se notas que é possível rodar browsers modernos, bem como renderizadores de imagens e PDF.

2.3 Comparações entre a performance e adoção de sistemas operacionais

2.3.1 Performance

Quando testado em termos de eficiência, o Windows 10 performa-se bem abaixo do Linux, em tarefas em nível de usuário (SULAIMAN; RAFFI, 2021).

Em um mesmo hardware, os programas que são executados no pano de fundo, continuamente, pelo Windows consumem um valor próximo de 5% de CPU e 41%

de RAM. Enquanto, no Linux Mint - uma versão popular de Linux - o consumo é de 1.8% de CPU e 24% de RAM. Uma diferença de performance de mais de mais de 200% em CPU e aproximatamente 200% em RAM (SULAIMAN; RAFFI, 2021).

A execução de um programa escrito em VBS - relacionado aos programas do pacote Office -, se dá com uma diferença de 0.501 segundos para o Linux Mint e 4.75 segundos no Windows. E.g., existe uma diferença absoluta de $\frac{4.75-0.501}{0.501} = 423\%$ em performance (SULAIMAN; RAFFI, 2021).

Por fim, também existem pesquisas feitas com rigor técnico em todas as outras área de manuseação de firmwares/harwares, como wireless (S.DEVAN, 2013); paralelismo e manuseação de aplicações em servidores (AVELEDA et al., 2010); programas científicos (Fast Fourier Transform) et al (D' ELIA; PACIELLO, 2011); performance em arquiteturas de Realidade Virtuais (VR) (THUBAASINI; RUSNIDA; ROHANI, 2010) etc. Todas elas demonstram uma performance superior na renderização e/ou no tempo de renderização em plataformas GNU/Linux (AVELEDA et al., 2010; THUBAASINI; RUSNIDA; ROHANI, 2010; S.DEVAN, 2013; SULAIMAN; RAFFI, 2021; D' ELIA; PACIELLO, 2011).

2.3.2 Demografia da adoção de FOSS

Fritzgerald enunciou em 2006 que o perfil de adoção dos sistemas open source estava numa fase "OSS 2.0", o qual transcendera a generalização de que isso era algo inutilizável pela pessoa média e uma ferramenta exclusiva de "hackers" (FITZGERALD, 2006). E, que ainda haveriam conseguintes transformações até um patamar mainstream. Isto é, se tornaria um lugar comum na sociedade e industria, e que grande parte do desenvolvimento seria de autoria de grandes empresas.

Em 2008, foi feita uma pesquisa sobre a adoção da indústria de softwares, na Finlândia. Em contraste à caracterização de Fritzgerald, 50% das empresas utilizavam amplamente de Open Source Software, porém, a participação era quase nula na comunidade aberta. Por fim, mais de 30% dessas empresas participantes relatavam que 40% do ganho era fruto de produtos e serviços desenvolvidos pela comunidade aberta(HAUGE; SØRENSEN; CONRADI, 2008). É importante se notar que eram empresas de pequeno e médio porte, em sua maioria.

Em 2012, um estudo em grande empresas - mil companias listadas na revista US Fortune - derivou algumas conclusões (SPINELLIS; GIANNIKAS, 2012):

- A adoção está diretamente associada à equipes de TI com trabalhos os quais demandam expertise e conhecimentos extensivos especializados, bem como busca por eficiência (GALLEGO et al., 2015; LI; TAN; YANG, 2013).
- Aumento exponencial, à partir de um ponto de acumulação;

- Existe um efeito-rede responsável pela adoção nas grandes. empresas;

Assim, a adoção em empresas de pequeno, médio e grande porte variam grandemente, e não possuem efeitos notáveis na expansão do movimento. No entanto, ainda são positivamente influenciadas por ele (SPINELLIS; GIANNIKAS, 2012; HAUGE; SØRENSEN; CONRADI, 2008; FITZGERALD, 2006). A maior influência positiva no movimento, indica-se, é dado por pequenas empresas (KSHETRI, 2004).

Observa-se no entando, de 2012 ao presente, uma intensificação de adoção e ubiquidade da adoção dessas tecnologias abertas (SCHMIDT, 2016). São consideradas, atualmente, o berço inovatório da indústria de softwares (SCHRAPE, 2019; SCHMIDT, 2016).

Pesquisas mais recentes determinam que, independente do tamanho da empresa, a adoção da empresa depende diretamente do treinamento dos agentes de TI e sua inclinação por autonomia (RACERO; BUENO; GALLEGO, 2020). Finalmente, a adoção e crença na efetividade desses softwares depende de fatores principalmente intrínsecos e independem de treinamento, por mais que os treinamentos em OSS potencializa a adoção de usuários inclinados ao uso (RACERO; BUENO; GALLEGO, 2021).

2.4 Demonstração de Aplicações na Indústria

2.4.1 Freqtrade

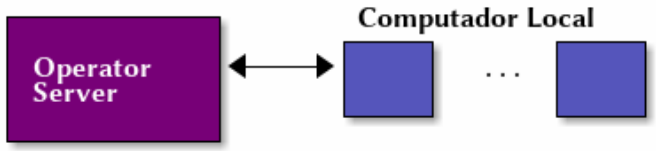
O software Freqtrade contém as peças essenciais para se rodar um aplicação de trocas-automatizadas entre crypto-moedas. Por meio de APIs, pode-se estender a aplicação para comunicar-se com o telegram. O programa foi escrito em Python e seu código é aberto à estensão, possuindo centro e quarenta e sete contruibuidores, com existência datando de Maio de 2017 (FANG et al., 2020).

O funcionamento das trocas-automatizadas se por meio de uma comunicação de mão dupla entre o computador local e um servidor-plataforma de trocas - a qual seria análoga a uma plataforma-corretora.

Grande parte do trabalho de se escrever um robô autômato, para qualquer fim, compraz em programar protocólogos de comunicação com servidores. Pois, o robô deverá ser capaz de dizer o servidor quais operações devem ocorrer, tanto gerando respostas ao cliente - dados a serem armazenados no computador local -, quanto operações internas ao servidor - como, executar uma compra e venda de criptomoeda.

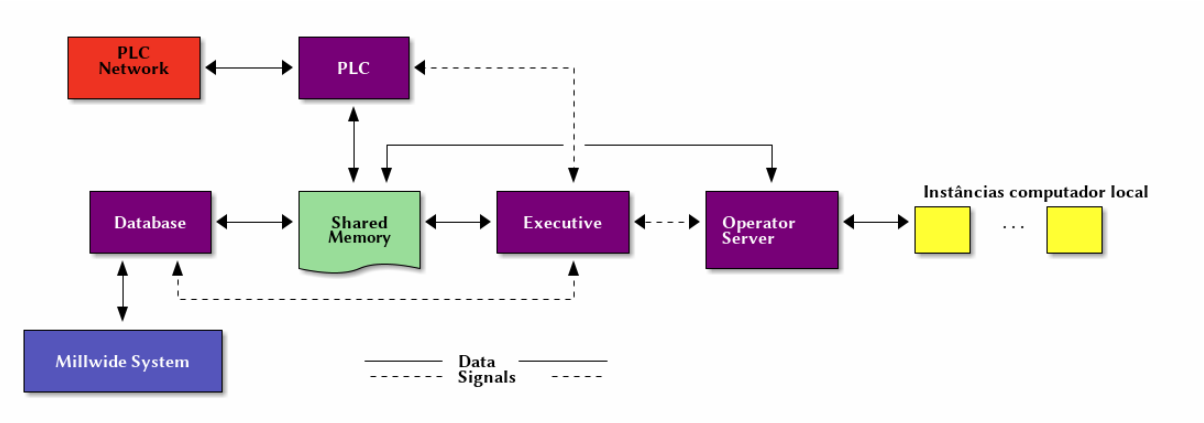
Assim, um diagrama completo o qual captura essa complexidade poderia es-

Figura 6 – Diagrama esquemático simplificado - comunicação cliente-servidor



Fonte: O autor.

Figura 7 – Diagrama esquemático - comunicação cliente-servidor



Fonte: O autor.

quematicamente ser visto na Figura 7.

Ao se utilizar o software aberto, toda essa complexidade da Figura 7 se torna mais próxima à Figura 6, pois toda a abstração-estrutural já se encontra escrita. Basta, para quem decidir utilizar o programa ler, entender e modificar o conteúdo disponível na documentação do projeto.

2.4.2 OR-Tools

A ferramenta OR-Tools traduz-se em bibliotecas largamente desenvolvida pela comunidade aberta, e a empresa Google (); utilizável em C++, Python, Java, (Clojure), C#, .Net.

OR-Tools possui a utilidade de resolver problemas como agendamento de horários, o qual, estruturalmente, assemelha-se aos algorítimos de solução de problemas como Sudoku ().

De forma geral, esses problemas caem dentro de um ramo da matemática e computação chamado Problemas de Optimazação Constrita (Constraint Optimization)

2.5 Demonstração de Aplicações Acadêmicas

2.5.1 DifferentialEquations.jl

A biblioteca numérica DifferentialEquations possui uma das melhores performances de softwares numéricos que existem (RACKAUCKAS; NIE, 2017b). Sua performace é comparativa à implementações em FORTRAN e C. Existem ports da biblioteca para Python e R, porém, desenvolveu-se em Julia.

Dentre a categoria de problemas possíveis de se resolver, utilizando-se ferramentas próprias do pacote, lista-se as seguintes categorias de equações diferenciais (RACKAUCKAS; NIE, 2019; RACKAUCKAS; NIE, 2017a; RACKAUCKAS; NIE, 2018; SYKORA et al., 2020; RACKAUCKAS et al., 2018; RACKAUCKAS et al., 2019; RACKAUCKAS et al., 2020; GOWDA et al., 2019; MA et al., 2021),

- Equações discretas (Estocásticos discretos (Gillepie/Markov), e mapas funcionais)
- Equações diferenciais ordinárias (ODEs)
- Equações estocásticas ordinárias (Integrações simpléticas, Método IMEX)
- Equações estocásticas diferenciais algébrigcas (SODEs or SDEs)
- Equações diferenciais aleatórias (RODEs ou RDE)
- Equações diferenciais algébricas (DAEs)
- Equações diferenciais com retardo (DDEs)
- Equações diferenciais neutras, retardadas e algebricamente retardadas (NDDES, RDDEs e DDAEs)
- Equações diferenciais estocásticas retardadas (SDDEs)
- Suporte parcial de soluções em equações diferenciais estocásticas neutras, retardads e retardadas algebraicas (SNDDEs, SRDDEs, SDDAEs)
- Equações mistas discretas e contínuas (Jump Diffusions)
- Equações diferenciais parciais (estocásticas) ((S)PDEs) - ambas com portabilidade de métodos finitos e/ou diferenças finitas)

2.5.1.1 Portabilidade em Julia

Para se utilizar do pacote, basta utilizar os seguintes comandos

```
# Carrega o gerenciador de pacotes, Pkg e instala, se preciso, o pacote.
using Pkg
Pkg.add("DifferentialEquations")
# Porta as utilidades do pacote, sem necessitar de referí-lo no meio do código
using DifferentialEquations
```

2.5.1.2 Portabilidade em Python

Num terminal, utilizando-se do gerenciador de pacotes pip,

```
pip install diffeqpy
```

Num terminal, rodando-se o interpretador de Python,

```
>>> import diffeqpy
>>> diffeqpy.install()
```

Por fim, opcionalmente, utilize numba para aumentar a performance do código,

```
pip install numba
```

Ademais, apenas repita o comando, dentro de um aquivo python,

```
import diffeqpy
```

2.5.1.3 Portabilidade em R

Para instalação,

```
install.packages("diffeqr")
```

Na primeira chamada de,

```
diffeqr::diffeq_setup()
```

Será feito o download do DifferentialEquations.jl. Um subconjunto específico do pacote é ativamente mantido no CRAN.

2.5.2 O LaTeX

O LaTeX possui separação entra as tarefas de produção de um documento. A linguagem permite-nos separar as tarefas de formatação do texto, da escrita de seu conteúdo. Desta forma, o usuário concentra-se exclusivamente em seu conteúdo, em um estágio da escrita do documento. E, na formatação de sua aparência, em outro momento.

Assim, ganha-se em qualidade de produção. Bem como, ganha total autonomia sob o documento, pois a programação da disposição gráfica dos elementos textuais pode ser programada - isto é, modificada indefinidamente, a partir dos comportamentos padrões dos pacotes utilizados. O sistema tipográfico de LaTeX chegou a ser considerado o sistema digital de tipografia mais sofisticado que existe, devido a essa paradigma de programação funcional, bottom-up (HARALAMBOUS, 2007).

O LaTeX, tecnicamente, é a junção do sistema de tipografia TeX, inventado por Donald Knuth, para tipografia de alto nível (KNUTH, 1986); com os poderosos macros que facilitam a extensão do programa TeX, a qual damos o nome de LaTeX. O LaTeX foi inicialmente desenvolvido por Leslie Lamport, com seus pacotes fundamentais de formatação (LAMPORT, 1994). O LaTeX, por conseguinte, não é somente uma linguagem de tipografia de alto nível, mas também um conjunto de macros para facilitar a tipografia em si. Qualifica-se, assim, como um sistema de preparação de documentos; uma linguagem markup de domínio específico.

2.5.2.1 Classe Canônica ABNT de produção científica

Documentos sob os requisitos das normas ABNT (Associação Brasileira de Normas Técnicas) para elaboração de documentos técnicos e científicos brasileiros - como artigos científicos, relatórios técnicos, trabalhos acadêmicos, como teses, dissertações, projetos de pesquisa e outros documentos do gênero (ABNTEX, 2012) - é ao que se chama classe canônica ABNT.

> Os documentos indicados tratam-se de “Modelos Canônicos”, ou seja, de modelos que não são específicos a nenhuma universidade ou instituição, mas que implementam exclusivamente os requisitos das normas da ABNT, Associação Brasileira de Normas Técnicas. (ARAUJO, 2018, Cap. 1)

As normas as quais prescrevem o modelo canônico são:

- **ABNT NBR 6022:2018:** Informação e documentação - Artigo em publicação periódica científica - Apresentação.
- **ABNT NBR 6023:2002:** Informação e documentação - Referência - Elaboração.
- **ABNT NBR 6024:2012:** Informação e documentação - Numeração progressiva das secções de um documento - Apresentação.
- **ABNT NBR 6027:2012:** Informação e documentação - Sumário - Apresentação.
- **ABNT NBR 6028:2003:** Informação e documentação - Resumo - Apresentação.

- **ABNT NBR 6029:2006:** Informação e documentação - Livros e folhetos - Apresentação.
- **ABNT NBR 6034:2004:** Informação e documentação - Índice - Apresentação.
- **ABNT NBR 10520:2002:** Informação e documentação - Citações.
- **ABNT NBR 10719:2015:** Informação e documentação - Relatórios técnicos e/ou científico - Apresentação.
- **ABNT NBR 14724:2011:** Informação e documentação - Trabalhos acadêmicos - Apresentação.
- **ABNT NBR 15287:2011:** Informação e documentação - Projeto de pesquisa - Apresentação.

2.6 Trabalhos Canônicos na Área de Computação

2.6.1 Structure and Interpretation of Classical Mechanics (SCIM)

A bilioteca científica de simulações de mecânica clássica, em Scheme (um dialeto de Lisp), foi escrito com intento de ser utilizado em cursos de mestrado no MIT. Acompanhado à biblioteca, existe o livro, o qual serve de material didáctico ao curso (SUSSMAN; WISDOM, 2015).

2.6.2 Structure and interpretation of computer programs (SICP)

O curso de Scheme (SICP), o qual é ensinado como materia básica de computação no MIT, possui como acompanhantes um dos mais influentes livros já escritos na história da computação (ABELSON; SUSSMAN, 1996). O curso é pioneiro em aprofundar-se epistemologia da programação. Onde, o uso de Scheme, o qual possui notação uniforme para todas as funcionalidades da língua foi revolucionário. Conquanto, em contraste aos outros cursos de fundamentos da computação focavam em ensinar a linguagem mais amplamente utilizada no momento.

O curso SCIM (subseção 2.6.1) foi um precursor de SICP. O curso continua ativo e o livro continua a ser utilizado amplamente no MIT, e em inúmeras outras faculdades. Uma lista não exaustiva pode ser encontrada em <https://mitpress.mit.edu/sites/default/files/sicp/adopt-list.html>, em que pelo menos, vinte universidades o adotam, em mais de oito países.

2.6.3 SICMUtils - Portabilidade de (SCIM) em Clojure

Existe uma biblioteca reescrita em clojure, a qual porta as funcionalidades descritas na biblioteca científica SCIM (SMITH; RITCHIE, 2016). Essa biblioteca é escrita numa das mais modernas linguages de computação com adoção amplamente da

Industria, Clojure. Recentemente, a maior empresa utilizado de Clojure até então, a Cognitect, a qual inventou a língua foi comparada pelo Nubank (HAMILTON, 2020).

A língua é tão poderosa que as implementações não são apenas numéricas, nem são somente simbólicas. É possível utilizar-se da mesma função (procedure) para se fazerem cálculos numéricos ou simbólicos, tendo apenas de se mudar os argumentos de entrada (inputs).

3 Materiais e Métodos

O planejamento da monografia seguiu as seguintes etapas:

1. Contato e estratégias com o orientador;
2. Determinação do tópico principal;
3. Delineação específica dos subtópicos;
4. Organização da agenda à partir das datas finais objetivadas;
5. Início à escrita do TCC;
6. Junção de notas exploratórias sobre o tópico;
7. Pesquisa bibliografia;
8. Escrita de todos os tópicos da monografia;
9. Revisão contínua à cada etapa da monografia;
10. Organização de apresentação;

3.1 Convite ao orientador

Procurou-se um docente tanto familiar ao autor, quanto versado e interessado no tópico de pesquisa.

3.2 Delineação do tema

Dado que o tema softwares livres na indústria e academia é um tópico indefinidamente abrangente, houve uma necessidade imprescindível de delimitar graduamente o tópico.

Nas revisões contínuas, determinava-se o quão profundo tinha se abordado os tópicos, e se era necessário dar procedência, ou seguir o andamento de outras parte da agenda.

3.3 Organização cronológica

Decidiu-se por seguir o método de diagrama de Gantt, amplamente utilizado em organização de projetos. As partições temporais foram ponderadas pelas datas limites objetivadas.

3.4 As fases da escrita da Monografia

Como a monografia se caracteriza como uma Análise Exploratória, o enfoque foi na literatura concernente ao Tema.

Na subtópico acadêmico, procurou-se dar enfoque ambos softwares auxiliares de pesquisas, quanto úteis à sala de aula. Bem como, pretendia-se explicar conceitos-chave na compreensão da dinâmica histórica e atual do ecossistema em que se desenvolve e aplicam softwares.

Ademais, nas aplicações tecnológicas, o autor documentou softwares que já utilizou em situações reais.

3.5 Junção de notas

Ao se utilizar os softwares os autores havia produzido anotações sobre o desenvolvimento do software. Uma auto-documentação, enquanto se desenvolvia, ou se estudava os mecanismos de uma aplicação. Assim, utilizou-se dessas notas para construção da monografia.

3.6 Pesquisa bibliográfica

A revisão da bibliografia objetiva facilitar o entendimento da indústria dos softwares, a qual é difere em seu aspecto majoritariamente livre e aberto. Além do mais, procurou-se fazer um recorte de história, usos e desenvolvimento em estado da arte de diversos softwares icônicos.

A maior quantidade de informações históricas encontra-se armazenada digitalmente, em sites de instituições internacionais, como o GNU. Conquanto, nas aplicações procurou-se o embasamento em artigos científicos.

Utilizou-se do Google Scholar e a plataforma da ACM (Association for Computing Machinery), da qual o autor é membro e na qual existe material especializado em computação e softwares.

3.7 Revisão contínua

Parte fundamental da organização da monografia foi a troca de informações, auxílio, e integração constante das considerações do orientador Dr. Wei-Liang.

3.8 Organização da apresentação

Na apresentação, procura-se utilizar ferramentas interativas e demonstração de conceitos por meio de programação em tempo real aplicações. Escolheu-se utilizar da ferramento Org-mode, integral ao Emacs.

4 Resultado e Discussões

4.1 Convite ao orientador

O autor já havia interagido com o Dr. Wei-Liang, o qual no passado havia pedido, a titulo de pesquisarem juntos, que se instalasse o sistema operacional do Linux. Ademais, uma das aplicações era na resolução e simulação de sistemas resolvidos por Lagrangeanas - tópico lecionado pelo orientador, na disciplina de Mecânica Clássica.

Assim, fortunamente o orientador se sentiu inclinado a participar desse trabalho. E, concorda que o ensino sobre a iniciativa Open Source, bem como os softwares em si, são detrimentais na vida acadêmica.

4.2 Delineação do tema

A iniciativa GNU e o open source é indefinidamente abrangente. Existem softwares abertos escritos para virtualmente qualquer outra área do exercício cognitivo do ser humano.

Necessitou contentar-se em parcialmente mostrar aplicações mais em mão, as quais seriam mais relevantes ao escopo da monografia. Bem como, optou-se por aprofundar-se em tópicos já utilizados na vida do autor para se resolver problemas na academia e na indústria.

Assim, a delimitação principal se deu entorno de ferramentas abertas das quais, na indústria foram apresentados OR-Tools e Freqtrade.

4.2.1 Da Indústria

As línguas utilizadas para o desenvolvimento desses softwares foram Python e C++. Porém, no caso de OR-Tools existe portabilidade para C++ (nativo), Python, Java, .NET. Todos os projetos e línguas possuem licença aberta.

OR-Tools destina-se a resolver um problema virtualmente conspícuo a qualquer negócio ou estrutura social prestadora de serviço. Pois, o problema de agendamento de horários, com condições de restrições impostas, é extremamente comum. Por exemplo, para organizar turnos de funcionários, em uma empresa; turnos de enfermeiros em um hospital - exemplo dado na própria documentação do software etc.

Freq-trade destina-se a automação de investimentos em crypto-moedas. A ferramenta surgiu espontaneamente da comunidade aberta, sem incentivo algum externo - até, pois, não existe um responsável que chefia o projeto. E, num curto período de desenvolvimento, cinco anos, desde 2017, a aplicação conta com inúmeras otimizações computacionais, e APIs com interfaces como Telegram. Não existe paralelo de software fechado o qual é comercializado, com mesmo escopo.

4.2.2 Da academia

Optou-se a estudar aplicações escritas em Python, Julia, Clojure(script). Pois, o autor possui histórico de utilização dessas ferramentas. Com efeito, a pesquisa constituiu, em grande parte, na junção de todos os esparços projetos em que o autor já havia colaborado ou desenvolvido. E, no aprofundamento da compreensão de todas esses projetos, para que fosse possível fazer uma apresentação coesa do ferramental utilizado.

Os programas em Julia possuiam cunho numérico. Pois, sua computação optimizada compara-se a performance de FORTRAN e C++. Assim, o principal pacote estudado foi DifferentialEquations.jl, a qual pode ser utilizada para resolver uma gama de equações diferenciais, incluindo a categoria Estocástica.

O programa em Python é de processamento de imagens. Escolheu-se essa aplicação, pois demonstra o quão abrangente é a portibilidade de programas escritos em outras línguas ao Python - principalmente, devido a grande comunidade de cientistas e programadores que o utilizam.

4.2.3 A intersecção

Com Clojure(Script), Bash e LaTeX, foi possível criar tanto simulações que estendem a compreensão de fenômenos físicos do campo abstrato ao visual. Também, empregou-se de Closh e Babashka, duas bibliotecas de Clojure dedicadas a interoperabilidade com Bash, a qual se processava milhares de arquivos, por meio de pipelines de arquitetura complexas. Assim, facilitando a escrita de uma quantidade enorme de relatórios técnicos na área de segurança do trabalho.

Quanto a Clojure(Script), foram pesquisadas simulações gráficas, as quais inerentemente são simples de se programar em Clojure. Pois, seu ecossistema pode interoperar entre JVM (Java Virtual Machine) e JavaScript. Assim, dado que ambos Java e JavaScript foram as línguas mais populares por décadas, existem uma enorme quantidade de ambientes e bibliotecas a nossa disposição. Bem como, Clojure é uma Lisp - e, caracteristicamente, essa família é utilizada para prototipagem, vide AutoCAD escrito em AutoLisp.

Dado que existem bibliotecas dedicadas a Mecânica Clássica em nível de mestrado, bem como exite capacidades de se renderizar conceitos físicos abstratos. Então, decorreu-se a computação de programas que demonstravam conceitos de Física, Cálculo, e Computação Generativa.

4.3 Organização cronológica

A organização do projeto se deu utilizando os diagramas de Gantt, e as agendas dinâmicas, /org-agenda/, nativas do Emacs. Por conseguinte, toda a comunicação sobre o projeto foi materializado por meio de cronogramas e agendas.

4.4 As fases da escrita da Monografia

Iniciou-se a escrita da monografia pela pesquisa bibliográfica, dado que a pesquisa se categoriza, epistemologicamente, como uma Análise Exploratória. O template utilizado para escrita do projeto foi o abnTEX2, biblioteca do LATEXdedicado a escrita de documentos sob as normas ABNT.

4.5 Junção de notas

O org-mode é uma ferramenta de organização e anotação, desenvolvida para pesquisa e reprodutibilidade de pesquisas. Assim, todas as etapas de desenvolvimento do projeto foi desenvolvido e documento utilizando essa ferramenta.

A documentação implícita ao desenvolvimento é uma característica fundamental e pensada do Org-mode. Pois, segue o paradigma de Literate Programming. Assim, há uma mistura de notas e códigos em um mesmo ambiente. Fazendo que a prototipação do código seja automaticamente documentável.

Por fim, é possível exportar qualquer anotação de extensão .org para .tex e consequentemente .pdf. Assim, reutilizar dessas anotações foi uma tarefa natural.

4.6 Pesquisa bibliográfica

A pesquisa bibliográfica foi em grande parte encontrada em sites sem autores. Pois, a documentação foi criada ou pela comunidade, ou por instituições, não unicamente publicada em formato de livro ou artigo.

Da parte acadêmica, como o pacote de resolução de equações diferenciais, em Júlia, encontrou-se vasta literatura. Até porque, a língua foi criada para resolver problemas computacionais tocantes principalmente a comunidade científica do MIT.

Por fim, algumas ferramentas apenas existem documentadas e contidas inteiramente em repositórios, como o Github. Foram os casos do OR-Tools e as simulações em Clojure. São publicações puramente eletrônicas.

4.7 Revisão contínua

Dentro do cronograma concebido para desenvolvimento da monografia, pôs-se metas e etapas de escrita do documento. E, estipulou-se os conteúdos necessários para cada nova versão do documento, aliado a uma data limite. Assim, o retorno do orientador ao orientado pode se dar mais frequentemente.

4.8 Organização da apresentação

Como se está utilizando de diversas ferramentas e extensões dentro do Emacs e do Org-mode, em particular, optou-se por utilizar uma de suas funcionalidades dedicada a apresentações.

5 Conclusão

Dentro das iniciativas Open Sourced, existem virtualmente aplicações para todas as áreas das aventuras do ser humano. Isto é, seja o uso de computação gráfica, processamento de imagens, expressões artísticas, ou ferramentas de organização, há um software escrito para esse fim.

É observável a falta de competição quanto a quantidade e qualidade de softwares fechados e abertos. Ademais, a capacidade de extensão dos softwares livres, por meio das quatro fundamentais liberdades de suas licenças, permitem que exista uma variedade imensa de uso de uma mesma ferramenta. Por fim, o enriquecimento dessa participação colaborativa solidifica e generaliza as ferramentas abertas - num grau e rapidez inimagináveis, comparados ao processo de desenvolvimento por uma equipe fechada.

Demonstrou-se diversos usos desses software, sem paralelos, na Indústria e na Academia. Ambos possuem algorítimos na fronteira da ciência, a qual o uso apenas é limitado pela imaginação e capacidade formativa do usuário-desenvolvedor. Assim, é imprescindível que no curso de engenharia se tenha uma ampla formação com base nos softwares livres, sob os quais a computação moderna subiste.

Apontou-se algumas portas de entrada, fundamentais, a esse paradigma, os sistemas com Kernel Linux, as aplicações GNU as quais lhe dão roupagem, e por fim os sistemas sociais onde se desenvolve os softwares, como a plataforma de versionamento Git/Github. Ademais, demonstrou-se, com efetividade, que é possível construir e rodas todas essas aplicações num sistema totalmente aberto.

Referências

ABELSON, H.; SUSSMAN, G. J. Structure and interpretation of computer programs. [S.l.]: The MIT Press, 1996. Citado 2 vezes nas páginas 6 e 19.

ABNTEX, E. A classe abntex2: Modelo canônico de trabalhos acadêmicos brasileiros compatível com as normas abnt nbr 14724: 2011, abnt nbr 6024: 2012 e outras.[sl], 2012. http://code. google. com/p-/abntex2/>. Citado, v. 2, p. 2, 2012. Citado na página 18.

AMIN, N.; ROMPF, T. Collapsing towers of interpreters. Proc. ACM Program. Lang., Association for Computing Machinery, New York, NY, USA, v. 2, n. POPL, dez. 2017. Disponível em: <https://doi.org/10.1145/3158140>. Citado na página 7.

ARAUJO, L. C. A classe abntex2. Documentos técnicos e científicos brasileiros, 2018. Citado na página 18.

AVELEDA, A. A. et al. Performance comparison of scientific applications on linux and windows hpc server clusters. Mecánica Computacional, v. 29, n. 30, p. 2985–2997, 2010. Citado na página 13.

D' ELIA, M.; PACIELLO, V. Performance evaluation of labview on linux ubuntu and window xp operating systems. In: IEEE. 2011 19thTelecommunications Forum (TELFOR) Proceedings of Papers. [S.l.], 2011. p. 1494–1498. Citado na página 13.

EMACS History. 2021. <https://www.emacswiki.org/emacs/EmacsHistory>. Citado na página 11.

FANG, F. et al. Cryptocurrency trading: a comprehensive survey. arXiv preprint arXiv:2003.11352, 2020. Citado na página 14.

FINK, M. The business and economics of Linux and open source. [S.l.]: Prentice Hall Professional, 2003. Citado na página 6.

FITZGERALD, B. The transformation of open source software. MIS quarterly, JSTOR, p. 587–598, 2006. Citado 2 vezes nas páginas 13 e 14.

GALLEGO, M. D. et al. Open source software: The effects of training on acceptance. Computers in Human Behavior, Elsevier, v. 49, p. 390–399, 2015. Citado 3 vezes nas páginas 8, 9 e 13.

GOWDA, S. et al. Sparsity programming: Automated sparsity-aware optimizations in differentiable programming. 2019. Citado na página 16.

HAMILTON, A. Brazilian challenger Nubank acquires firm behind Clojure and Datomic. 2020. <https://www.fintechfutures.com/2020/07/brazilian-challenger-nubank-acquires-firm-behind-clojure-and-datomic/>. Citado na página 19.

HARALAMBOUS, Y. Fonts & encodings. [S.l.]: " O' Reilly Media, Inc.", 2007. Citado na página 17.

HAUGE, Ø.; SØRENSEN, C.-F.; CONRADI, R. Adoption of open source in the software industry. In: SPRINGER. IFIP International Conference on Open Source Systems. [S.l.], 2008. p. 211–221. Citado 3 vezes nas páginas 9, 13 e 14.

HERTEL, G.; NIEDNER, S.; HERRMANN, S. Motivation of software developers in open source projects: an internet-based survey of contributors to the linux kernel. Research policy, Elsevier, v. 32, n. 7, p. 1159–1177, 2003. Citado na página 6.

HIPPEL, E. v.; KROGH, G. v. Open source software and the "private-collective" innovation model: Issues for organization science. Organization science, Informs, v. 14, n. 2, p. 209–223, 2003. Citado 2 vezes nas páginas 5 e 6.

KNUTH, D. E. The art of computer programming, vol 1: Fundamental. algorithms, p. 187, 1968. Citado na página 5.

KNUTH, D. E. TEX: the Program. [S.l.]: Addison-Wesley, 1986. Citado na página 18.

KSHETRI, N. Economics of linux adoption in developing countries. IEEE software, IEEE, v. 21, n. 1, p. 74–81, 2004. Citado na página 14.

LAMPORT, L. LATEX: a document preparation system: user' s guide and reference manual. [S.l.]: Addison-wesley, 1994. Citado na página 18.

LI, Y.; TAN, C.-H.; YANG, X. It is all about what we have: A discriminant analysis of organizations' decision to adopt open source software. Decision support systems, Elsevier, v. 56, p. 56–62, 2013. Citado 3 vezes nas páginas 8, 9 e 13.

LINUX, list of distributions. [S.l.]: Wikimedia Foundation, 2021. <https://en.wikipedia.org/wiki/List_of_Linux_distributions>. Citado na página 11.

MA, Y. et al. ModelingToolkit: A Composable Graph Transformation System For Equation-Based Modeling. 2021. Citado na página 16.

MACOS version history. [S.l.]: Wikimedia Foundation, 2021. <https://en.wikipedia.org/wiki/MacOS_version_history>. Citado na página 10.

MICROSOFT, list of operating systems. [S.l.]: Wikimedia Foundation, 2021. <https://en.wikipedia.org/wiki/List_of_Microsoft_operating_systems>. Citado na página 10.

MIROWSKI, P. The future (s) of open science. Social studies of science, SAGE Publications Sage UK: London, England, v. 48, n. 2, p. 171–203, 2018. Citado na página 5.

MOODY, G. Rebel code: Linux and the open source revolution. [S.l.]: Hachette UK, 2009. Citado na página 6.

PETERS, M. A. Open education and the open science economy. Yearbook of the National Society for the Study of Education, v. 108, n. 2, p. 203–225, 2009. Citado 2 vezes nas páginas 5 e 6.

RACERO, F. J.; BUENO, S.; GALLEGO, M. D. Predicting students' behavioral intention to use open source software: A combined view of the technology acceptance model and self-determination theory. Applied Sciences, Multidisciplinary Digital Publishing Institute, v. 10, n. 8, p. 2711, 2020. Citado 3 vezes nas páginas 8, 9 e 14.

RACERO, F. J.; BUENO, S.; GALLEGO, M. D. Can the oss-focused education impact on oss implementations in companies? a motivational answer through a delphi-based consensus study. Electronics, Multidisciplinary Digital Publishing Institute, v. 10, n. 3, p. 277, 2021. Citado 2 vezes nas páginas 8 e 14.

RACKAUCKAS, C. et al. Diffeqflux.jl-a julia library for neural differential equations. arXiv preprint arXiv:1902.02376, 2019. Citado na página 16.

RACKAUCKAS, C. et al. A comparison of automatic differentiation and continuous sensitivity analysis for derivatives of differential equation solutions. arXiv preprint arXiv:1812.01892, 2018. Citado na página 16.

RACKAUCKAS, C. et al. Universal differential equations for scientific machine learning. arXiv preprint arXiv:2001.04385, 2020. Citado na página 16.

RACKAUCKAS, C.; NIE, Q. Adaptive methods for stochastic differential equations via natural embeddings and rejection sampling with memory. Discrete and continuous dynamical systems. Series B, NIH Public Access, v. 22, n. 7, p. 2731, 2017. Citado na página 16.

RACKAUCKAS, C.; NIE, Q. Differentialequations.jl–a performant and feature-rich ecosystem for solving differential equations in julia. Journal of Open Research Software, Ubiquity Press, v. 5, n. 1, 2017. Citado na página 16.

RACKAUCKAS, C.; NIE, Q. Stability-Optimized High Order Methods and Stiffness Detection for Pathwise Stiff Stochastic Differential Equations. arXiv:1804.04344 [math], 2018. Disponível em: <http://arxiv.org/abs/1804.04344>. Citado na página 16.

RACKAUCKAS, C.; NIE, Q. Confederated modular differential equation apis for accelerated algorithm development and benchmarking. Advances in Engineering Software, Elsevier, v. 132, p. 1–6, 2019. Citado na página 16.

SCHMIDT, C. Agile software development. In: Agile Software Development Teams. [S.l.]: Springer, 2016. p. 7–35. Citado na página 14.

SCHRAPE, J.-F. Open-source projects as incubators of innovation: From niche phenomenon to integral part of the industry. Convergence, SAGE Publications Sage UK: London, England, v. 25, n. 3, p. 409–427, 2019. Citado 2 vezes nas páginas 9 e 14.

S.DEVAN, S. Windows 8 v/s linux ubuntu 12.10 - comparison of the network performance. International Journal of Research in Engineering and Technology, v. 02, p. 577–580, 2013. Citado na página 13.

SMITH, C.; RITCHIE, S. SICMUtils: Functional Computer Algebra in Clojure. 2016. <http://github.com/sicmutils/sicmutils>. Citado na página 19.

SPINELLIS, D.; GIANNIKAS, V. Organizational adoption of open source software. Journal of Systems and Software, Elsevier, v. 85, n. 3, p. 666–682, 2012. Citado 3 vezes nas páginas 8, 13 e 14.

STALLMAN, R. Linux and GNU - GNU Project - Free Software Foundation. 1997. <https://www.gnu.org/gnu/linux-and-gnu.html>. Citado na página 11.

STALLMAN, R. My lisp experiences and the development of gnu emacs (transcript of richard stallman’ s speech, 28 oct 2002, at the international lisp conference). URL (consulted December 2003): http://www. gnu. org/gnu/rms-lisp. html, 2002. Citado na página 11.

SULAIMAN, N. S.; RAFFI, A. S. H. A. Comparison of operating system performance between windows 10 and linux mint. International Journal of Synergy in Engineering and Technology, v. 2, n. 1, p. 92–102, 2021. Citado 3 vezes nas páginas 8, 12 e 13.

SUSSMAN, G. J.; WISDOM, J. Structure and interpretation of classical mechanics. [S.l.]: The MIT Press, 2015. Citado na página 19.

SYKORA, H. T. et al. Stochasticdelaydiffeq. jl-an integrator interface for stochastic delay differential equations in julia. 2020. Citado na página 16.

TANENBAUM, A. S.; BOS, H. Modern operating systems. [S.l.]: Pearson, 2015. Citado na página 8.

THUBAASINI, P.; RUSNIDA, R.; ROHANI, S. Efficient comparison between windows and linux platform applicable in a virtual architectural walkthrough application. In: Innovations in Computing Sciences and Software Engineering. [S.l.]: Springer, 2010. p. 337–342. Citado na página 13.

TORVALDS, L. Linux: a portable operating system. Master’ s thesis, University of Helsinki, 1997. Citado na página 11.

TU, Q. et al. Evolution in open source software: A case study. In: IEEE. Proceedings 2000 International Conference on Software Maintenance. [S.l.], 2000. p. 131–142. Citado na página 6.

WEST, J.; DEDRICK, J. Open source standardization: the rise of linux in the network era. Knowledge, Technology & Policy, Springer, v. 14, n. 2, p. 88–112, 2001. Citado na página 6.