energies

Article

The Role of Open-Source Software in the Energy Sector

Jonathan Klimt [1,*], Niklas Eiling [1], Felix Wege [1], Jonas Baude [1] and Antonello Monti [1,2,*]

[1] Institute for Automation of Complex Power Systems, RWTH Aachen University, 52074 Aachen, Germany; niklas.eiling@eonerc.rwth-aachen.de (N.E.); fwege@eonerc.rwth-aachen.de (F.W.); jonas.baude@rwth-aachen.de (J.B.)
[2] Fraunhofer Institute for Applied Information Technology FIT, Digital Energy, 53757 Sankt Augustin, Germany
* Correspondence: jonathan.klimt@eonerc.rwth-aachen.de (J.K.); amonti@eonerc.rwth-aachen.de (A.M.)

Abstract: Fast digitalization of the power grids and the adoption of innovative software solutions is key to a successful energy transition. In other sectors, such as telecommunication or cloud computing, open-source software has already proven capable of transforming entire industries, by speeding up development and lowering development costs while achieving high levels of stability, interoperability, and security. However, the energy sector has not yet embraced open-source software to the same level. We discuss how existing open-source software principles can be applied to the unique challenges of the energy sector during the transition towards higher penetration of renewable energy resources. To provide an overview of the current state of the open-source software landscape, we collected and analyzed 388 open-source projects, in terms of project activities, community composition, relevant licenses, and commonly used programming languages. One finding was that the majority of projects are currently driven by academic contributors, but that commercial players do also play a role, and we identify positive examples of collaboration between the two, mostly related to standardization.

Keywords: open-source software; energy transition; power systems; free software; FOSS; licenses; community; future distribution grid

check for updates

Citation: Klimt, J.; Eiling, N.; Wege, F.; Baude, J.; Monti, A. The Role of Open-Source Software in the Energy Sector. *Energies* **2023**, *16*, 5855. https://doi.org/10.3390/en16165855

Academic Editor: Abu-Siada Ahmed

Received: 10 July 2023
Revised: 31 July 2023
Accepted: 3 August 2023
Published: 8 August 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Today, open-source software that can be freely shared, modified, and reused plays an important role in software development. Web servers, databases, protocols, and data format libraries are examples of components where open-source software is dominating the landscape, even as part of proprietary products.

The use of an open-source license can massively increase the adoption, use, and re-use of software, allowing developers to focus on the implementation of core features instead of duplicating many software components. The resulting reduction of development resources can then lead to speeding-up of the innovation process, faster time to market, and reduced R&D costs [1,2]. Other benefits claimed for open-source software are the minimization of the risk of market domination by single vendors or of limitation to certain geographical regions [3].

At the same time, the energy sector is currently transitioning from fossil-based energy generation to renewable energy sources in many countries, to accomplish the goals of the 2015 Paris Agreement [4] and similar climate goals, e.g., the EU's "fit for 55 package" to ensure 55% reduction (compared to 1990) of greenhouse gas emissions by 2030. However, moving towards renewable energy generation also implies the transition from centralized to distributed power generation, which poses challenges, such as system stability, voltage control, supply reliability, and equipment control, to the grid operators [5,6]. The upcoming electrification of the mobility and heating sectors will also affect grids, by creating new power profiles and higher levels of power demand [5,7]. Furthermore, new stakeholders, such as energy aggregators, local energy communities, or customers offering flexibility to the market, are entering the business value chain [8].

New power grid architectures demand full digitalization of operation, including system management, energy trading, sector coupling, and network planning. Furthermore, digital technologies, such as smart charging, demand-side management, or grid-aware consumers, offer great potential for increased efficiency and savings [9].

We argue that open-source software and its collaborative software development can facilitate the digitalization of energy systems and the development of new grid solutions. Embracing open-source software is one means of fostering the free exchange of knowledge and data, to achieve the ambitious goals of energy transition, and we observe a growing momentum in this regard.

A more recent example of this is the establishment of Linux Foundation Energy (LFE), a non-profit organization supporting open-source projects in the energy sector, by system operators, universities, and software vendors [10]. This indicates that the energy sector is starting to become aware of the opportunities of open-source software adoption.

However, the energy sector has not yet fully embraced open-source software, due to the momentum of established software products that arose in the era of centralized grids, when collaboration and interoperability were of less importance than they are now. Furthermore, due to the safety criticality of many products in the energy sector, the field is very conservative, thus resisting fast change. Concerns about security and losing competitive advantage lead to stakeholders being opposed to publishing any data connected to their infrastructure.

In this paper, we show that the advantages of open-source software—e.g., in fostering collaboration and innovation—outweigh these doubts.

We wanted to provide an overview of open-source software to the energy community, and to discuss why it can facilitate energy transition. We support the decision making process of choosing the correct strategy for creating successful open-source software by presenting licensing, community concepts, business models, benefits, and concerns regarding open-source software in the energy sector (Section 2).

As a second contribution, we gauge the current state of the open source landscape in this sector, by analyzing a data set of 388 open-source software projects related to power grids and electric energy distribution (Section 3).

2. Open-Source Software

Open-source software is characterized by a license that allows modification and free redistribution of the corresponding source code. In this first part of the paper, we discuss various aspects of this kind of software, with an emphasis on the implications for the energy sector.

2.1. Licensing

Early software, originating in academia in the 1960s, was typically shared freely without using licenses, so as to use limited resources efficiently and to facilitate knowledge exchange. With broader adoption in industry and academia, restricting the use of software and charging fees for each distributed copy became more common [11]. A countermovement advocating the benefits of unrestricted software led to the formalization of the two terms free software and open-source software. While these terms are defined by the Open Source Initiative [12] and the Free Software Foundation [13], respectively, both definitions show only minor differences [14].

Open-source software is software that may be used with very few restrictions, in terms of use and redistribution. The definitions of the Free Software Foundation and the Open Source Initiative for open-source software require granting users wide-ranging rights: the use of the software must be allowed without restrictions for a specific duration, geography, field, product, or group of people. Sharing of the software must be possible without additional fees. Modification and improvement of the software must be enabled by making the original source code available. If not all of these freedoms are granted, the software is not open-source software, i.e., it is proprietary software. Notably, this definition does not prohibit commercial use of open-source software, and it even allows selling the software, as long as redistribution and sharing are not restricted.

If not explicitly transferred, all rights to software lie exclusively with the copyright owner. This is commonly the author or an organization employing the author. The copyright owner can grant rights to the software by means of a license that regulates its use and sharing. The Free Software Foundation and the Open Source Initiative maintain lists of specific licenses they recognize as open-source licenses because they grant all rights required by the definition of open-source software [15,16]. When a copyright owner adds one of these licenses to software, it becomes open-source software. A good practice for new projects is to choose common licenses, in order to avoid unforeseen conflicts when combining projects with different licenses, or to avoid confusion about the actual rights granted. Table 1 shows a list of licenses the Open Source Initiative recommends being used for creating open-source software.

Table 1. Licenses recommended by the Open Source Initiative [12].

License	Copyleft (Strong)	Copyleft (Weak)	Permissive
Apache License 2.0			✓
3- or 2-clause BSD license			✓
GNU General Public License	✓		
GNU Lesser General Public License		✓	
MIT license			✓
Mozilla Public License 2.0		✓	
CDDL License 1.0		✓	
Eclipse Public License 2.0		✓	

The general agreement on the basic rights and freedoms attached to open-source software shows a wide consensus on what qualifies as open-source software. However, while the requirements of open-source software are strict, they still allow for license differences. Most notably, common licenses for open-source software differ in how they regulate the creation of derived work, i.e., if work based on pre-existing open-source software is also open-source software.

Copyleft licenses require derived versions that use parts of an open-source software to also be published under the same license as the original work: this way, copyleft licenses guarantee that the rights of open-source software are never removed [12,13]. A popular example of a copyleft license is the GNU General Public License (GPL), which stipulates that all derivative work must also be licensed using the GPL, without imposing additional restrictions. The GPL is a form of strong copyleft, where no derivative work under other licenses is allowed. In contrast, weak copyleft licenses allow some use under other licenses: for example, the GNU Lesser General Public License (LGPL) allows integration into software with other licenses, as long as the original work is not modified.

In contrast to copyleft licenses, permissive licenses do not limit the redistribution of software. This allows license changes for all derivative works, thereby enabling software with a permissive license to be reused in proprietary software [12,13]. An example of a permissive license is the Apache License 2.0, which only requires the copyright and license text to remain intact.

2.2. Communities

Successful open-source software projects are often driven by an extensive community [17,18]. Being open to contributions from anyone, open source communities are usually dynamic groups, featuring members with different motivations, skills, roles, degrees of involvement, and responsibility for the project. Thus, open source communities benefit from diversity that fosters innovation and enables high flexibility and fast reaction times, e.g., to security issues or new technologies. The members are usually organized in a hierarchy, with a multitude of roles and degrees of involvement [17,18]. The inner circle of community members have managerial roles and are responsible for accepting changes to the code base. Additionally, there are members with varying activeness who contribute to the project by fixing bugs, adding new features, or preparing documentation. By contrast, users do not contribute to the code base, but participate in the community in other ways, e.g., by supporting users, reporting bugs, testing features or providing other forms of feedback.

While many open source communities are self-administrated, some are backed by non-profit organizations or companies seeking commercialization of the open-source software [19]. Most projects that are community-initiated are self-administrated, i.e., are managed by one or more of the contributors who work on the project unrelated to an employment context. While sponsorship of companies is sometimes welcome, most self-administrated communities try to prevent too much influence by commercial entities, thereby staying independent but requiring the continued involvement of volunteers.

As open source communities grow, so do their administrative needs. To deal with this, several non-profit organizations have appeared that are dedicated to the building and supporting of open source communities. They offer a wide range of resources and services, including legal representation and advice, financial management, training, and community events [20]. In the context of open-source software for the energy sector, four non-profit organizations supporting communities, standardization, and networking are noteworthy: Linux Foundation Energy [10] (LF Energy) brings together different stakeholders from the energy sector; the FIWARE Foundation [21] drives the definition and adoption of open standards based on the FIWARE framework for open-source platform components in different domains, including smart energy and smart cities; the Grid Protection Alliance (GPA) focuses on the development and support of software solutions for the electricity sector; the OpenADR Alliance [22] addresses standardization and automation of Demand Response and Distributed Energy Resources. While with non-profit organizations some independence is still retained by the community, projects backed by companies are driven by commercial interests, and a significant proportion of contributions are often financed by one or more companies. This structure can bring long-term resources, provide legitimacy, and guarantee continuous development, while still inviting external contributions. However, the commercial interests of profit-seeking companies may conflict with the visions and goals of external contributors. When companies promoting open-source projects do not spend considerable effort on community building, they risk low levels of external participation. Deciding on adequate community governance depends on the open source strategy of a company.

The Linux Foundation stands out in this group of organizations, because it did not originate from the energy domain. The Linux Foundation hosts numerous projects that have transformed industry in other domains, such as automobile, financial services, and telecommunications. Two examples of the disruptive software hosted in LF are the Linux kernel and Kubernetes. In 2018, the LF Energy Foundation was launched, to speed up the technological transformation of power systems: since then, LF Energy has steadily increased the number of members, projects, and users of the open-source software hosted by the Foundation. The technical work of LF Energy is transparent and is open towards the participation of external contributors. Joining technical steering committees and working groups is not tied to membership in the Foundation. Each project's development is organized by a technical steering committee, where responsibilities are based on technical contributions.

2.3. Commercialization

Although open-source licenses grant wide-ranging rights to anyone free of charge, commercialization of open-source software or of product based on open-source software is still possible. Basing commercial software on existing open-source software is a frequently employed strategy that reduces development cost through code re-use [1]. The licensing of open-source software plays an important part in how commercialization can occur in these circumstances. When new software is built using existing open-source code with permissive licenses, the new software may have a proprietary license, and it can thereby be commercialized traditionally, by selling copies or subscriptions. This is not possible when the software itself has an open-source license, e.g., because it uses existing open-source code with copyleft licenses that require derived work to be also open-source. For open-source software, sharing cannot be restricted, making receiving royalties incompatible with open-source licenses.

However, various business models exist that focus on selling additional values around the core software itself [23]. Selling services—such as support, maintenance, or training—additional to the software product is a business model, which companies like RedHat successfully use. Similarly, deployment of open-source software can be offered in a subscription model, as software-as-a-service: in this case, the product consists of hosting and maintaining the service for customers, guaranteeing availability and functionality, thereby removing the need for the customer to provide and maintain their own infrastructure required by the software. In the Open Core business model, a software vendor offers core functionality under an open-source license (often referred to as a community edition), while offering additional feature options with proprietary licenses requiring payment. In the Software Certification business model, the software is available under an open-source license, but service providers or Original Equipment Manufacturers (OEMs) are obliged to undergo a certification procedure, in order to prove interoperability or to use a trademark. The Commercial Open-Source Software Company Index [24] lists 50 companies, each generating a revenue of at least USD 100 million based on open-source business models. Of these 50 companies, 45 are using the open-core business model.

2.4. Implications for the Energy Sector

Open-source software has the potential to transform entire industries, e.g., cloud computing, web infrastructure, big-data analytics, embedded systems, mobile devices, telecommunications, and the internet of things, with projects such as Linux, Hadoop, and Kubernetes, to mention just a few [1,3]. However, open-source software is not yet common in the energy sector. In this section, we discuss potential benefits of and objections to open-source software, and we link these to implications for the energy sector.

2.4.1. Interoperability

The proliferation of renewable energy resources leads to an increasing need for coordination among stakeholders in the energy grid. Effective automation, protection, and monitoring in a power system composed of a large amount of distributed intelligence requires interaction between vastly different types of software of various vendors and the use of a multitude of interfaces and data formats. This need for interaction requires that the components of the energy system are designed to be interoperable on all levels, ranging from hardware-level interoperability, through semantic interoperability for data processing in services, to organizational interoperability, for seamless alignment and integration of business processes beyond organizational boundaries. Achieving interoperability is a key challenge for future energy systems, because of the lack of inherent motivation in vendors and system operators to implement interfaces that enable interaction with components from competitors. Interoperability cannot be solved by a monolithic solution, but requires a new level of cooperation that takes security, safety, and data sovereignty between all stakeholders into account.

In many fields, interoperability between components has been implemented by means of standardization [25]. This has also been true for the energy domain, where the need for safety and security to ensure grid availability adds another motivation for standardization, to reduce overall system complexity. Here, specification-based standards driven by standards-developing organizations are common. However, developing and implementing standards has proven to be a slow and arduous process, dominated by large players that have low motivation for innovation [3]. In the face of ambitious emission and efficiency targets, we cannot solely rely on standardization to drive the implementation of industry-wide interoperability. Furthermore, interoperability is also becoming a more complex concept. In a digital energy scenario, interoperability is not only about data exchanges and protocols but also the interoperability of services. By adopting a Smart Grid Architecture Model (SGAM) approach [26], we could say that interoperability requirements are moving from the lower layers to the top layers, including the business level.

An open-source software approach brings interoperability to the center. Here, the combination of collaboration and contribution within an open community is an early focus [27]. This leads to more innovation and higher adoptions, which can in turn lead to the standardization of proven and vendor-neutral technologies. Open-source software facilitates code re-use, code inspection, and data translations, allowing the establishment of open interfaces and data formats even without standardization: this way, the various software products in the energy domain can be combined, data can be freely exchanged, and previously impossible services can be offered. As this openness leads to interoperability of software not being owned by a single entity, open-source software is an effective countermeasure to companies leveraging established proprietary software products to ensure market superiority.

One example of a successful demonstration of the benefits of open-source software as an enabler for interoperability is the Service-based Open-source Grid automation platform for Network Operation of the future (SOGNO) [28], an interoperability layer for energy system monitoring and automation. In contrast to monolithic solutions, SOGNO follows a modular, plug-and-play model of microservices that interact via open interfaces, to perform a variety of monitoring and automation tasks. In this way, SOGNO acts as an interoperability layer that connects various independent components. By using open-source software, any party can develop or improve components for this platform, thus being able to interact with any existing component. This is, for instance, interesting for grid operators, as SCADA systems are often very long-running systems. The integration of new components would be dependent on the software vendor, who might already be out of business or have a commercial conflict of interest. The open-source and open-interface approach avoids this and opens up the market for third parties.

Open-source platforms, such as SOGNO, can foster collaboration between industry and research, as open-source tools or service implementations can be optimized directly or enhanced by academics, and vice versa: this way, the transition from research to production becomes seamless.

Even energy markets, where participants place bids based on secret algorithms and data, can benefit from embracing open-source software.

While publishing details about internal decision making can reduce profits derived from having better insights than one's competitors, using open-source software does not prohibit keeping sovereignty over these details. Making the software infrastructure used by possibly proprietary decision algorithms interoperable and open-source can lead to more robust, flexible, and powerful products. Building on such software increases innovation, by enabling better access to market bidding infrastructure.

Lastly, the adoption of open-source solutions does not mean avoiding standardization. The claim is that open source can support a faster track towards a standard, based on a bottom-up approach, having a de facto standard agreed on in practice before the actual standardization effort starts.

2.4.2. Impact of Open-Source Software on Security and Safety

While software security and safety are multi-layered problems, there is a decades-old debate on the impact of the availability of the source code on the security of software systems. There is the argument that the availability of the code makes it easier for attackers to study the system, and thus, that it reduces the security. The principle of hiding the code is, hence, known as "Security by obscurity": the proponents of open-source software argue that this is a weak security principle and that true security can only be achieved by the absence of weaknesses [29,30]. The availability of the code to the public can help with this, as there are more people who can analyze the code and help find and fix such vulnerabilities. Ruffin and Ebert [31] claim that security breaches in open-source projects are usually fixed quickly. Payne [32] argues that this can happen even faster in open-source projects if there are no company patch schedules blocking the process. In any case, the presence of the source code only helps to improve the security of a software component if it is actually reviewed and checked for security vulnerabilities. It can be easily argued that such security audits of a whole software stack are easier to perform if the code is disclosed, as there is no dependency on the collaboration of the supplier of the code.

An important tool to leverage this advantage is the software bill of materials (SBOM), which describes the open-source and proprietary software dependencies of a software package. Both functional and security tests require software tools and infrastructure to run these tests. These resources can be shared between open-source projects: one example is the Linux Foundation, which runs the LFX platform and, in particular, LFX security [33]. Sharing data about security vulnerabilities facilitates the tracking of vulnerabilities across several software packages. Furthermore, the open-source nature of the code base allows for examination, to ensure the absence of bugs, viruses, and other security issues. This process can also be automated, to create security reports that identify known vulnerabilities in a project. While this alone does not make the software more secure, it makes the security of the code base transparent to the users. In any case, an important assumption is that open-source software development is performed by implementing strict quality standards. This is possible within organized communities, such as Linux Foundation Energy, where the governance makes sure that all the projects follow clear guidelines, to improve quality and, correspondingly, security.

2.4.3. Teaching

The increasing availability of open-source software has also enabled the wider use of energy-specific software, e.g., modeling or simulation tools, in the educational sector [34]. The open-source nature of such projects not only makes the software freely available for universities and students, but also provides deep technical insights into the implementation of tools and algorithms in lectures or practical laboratory sessions, which supports capacity building in academia and industry, as students gain experience with sector-specific tools. The latter is also in accordance with the perception that open-source software helps to reduce training time for new employees or graduates, as it increases the likelihood of previous experience with the tools [1].

2.5. Objections and Risks

Despite the mentioned benefits, some stakeholders have raised objections to the adoption of open-source software. While there are several business models that propose ways of monetizing open-source software, a fear of not being able to compete with proprietary software, due to a lack of financial resources, persists [17,35].

However, there are numerous examples of companies generating substantial revenue based on open-source software [24]. In Section 2.3, we discuss various strategies to achieve commercial success with open-source business models.

Open-sourcing an existing software code base always carries the risk of undesirable exposure. Flaws in the code could decrease the credibility of the research [2] or product. Hence, companies may fear substantial costs for reviewing and potentially refactoring the code base prior to its publication.

Forking is a risk for open-source projects, because it leads to community splitting, which can induce the decay of the community and can lead to unmaintained code bases [17]. This risk of abandoned projects also needs to be considered when building proprietary software based on open-source projects. Community governance strategies can reduce the danger of forking for a project but cannot prevent it entirely. Depending on the applicable license, e.g., some permissive licenses, forks can even continue under a proprietary license [17].

Furthermore, the large number of licenses to choose from is sometimes criticized: it leads to high complexity in choosing an adequate license, and some developers are concerned about the lack of clarity in open-source licenses, e.g., regarding definitions of derived work or liability [17]. To remedy this, in Section 2.1, we give an overview of the most popular open-source licenses and the consequences of their use.

In any case, it should be underlined that the strict opposition between proprietary and open-source software is also a limited view. Many of the accepted licensing models of open source are based on the assumption of coexistence with proprietary solutions. This sort of compromise represents a very interesting business model, in which open source brings significant cost reduction also to companies operating with proprietary solutions. The right level of cooperation allows companies to focus on unique selling propositions, while leveraging open solutions in the background. This is an interesting compromise that also goes in the direction of interoperability, as discussed in Section 2.4.1.

3. Analysis of Open-Source Software Projects in the Energy Sector

To analyze the current state of the open source landscape in the energy sector, we created an extensive open-source software catalog containing various open-source projects in the energy sector. We collected a data set of 388 projects, by reviewing online resources and literature and by interviewing professionals involved in software development in the energy sector. The software was categorized into electrical-grid-related categories (Figure 1) and other energy-sector-relevant categories (Figure 2). However, this classification can be ambiguous, as, for instance, some grid-simulation software also includes aspects of modeling. In this section, we will use this set to analyze the community composition, the license distribution, the programming language distribution, and the project activity over time.

3.1. Related Work

We were interested in the current state of the open source landscape in the energy domain. To get a holistic image, we researched the literature and online resources for catalogs and collections of such software. This revealed multiple collections of such software: however, many of these did not focus on open-source software [36] or they analyzed only some areas of the energy sector, e.g., optimization [37] or PV modeling [38,39]. There are also multiple collections of open-source software projects in the energy sector outside scientific literature [10,12,40,41].

Groissböck [37] reviewed 31 open-source and proprietary tools for energy system modeling and optimization and proposed a maturity analysis based on the implemented features of the software. The author checked the selected software for 81 features commonly required in the optimization field. Using this metric, the paper showed that some considered open-source software tools can compete with proprietary tools. The paper concluded that most of the considered tools have a specific focus instead of aiming for a wide feature set, with open-source tools having a smaller feature set in general. While the set of supported features is important for monolithic proprietary software, we find that highly specialized software does limit applicability less for open-source software: this is due to the possibilities of open-source software for code re-use and consolidation into a feature-rich platform that makes even such specialized software applicable to abstract problems.

Holmgren et al. [38] reviewed existing open-source software tools for PV modeling, and they discussed funding schemes for such tools. They provided a list of 16 tools and libraries, comparing their licenses, programming languages, and purposes. The authors published the list in their paper. By focusing on PV modeling tools, the scope of the list was limited and, although the list was also published on a website, it has not been kept up to date, therefore representing a fixed view of 2018.

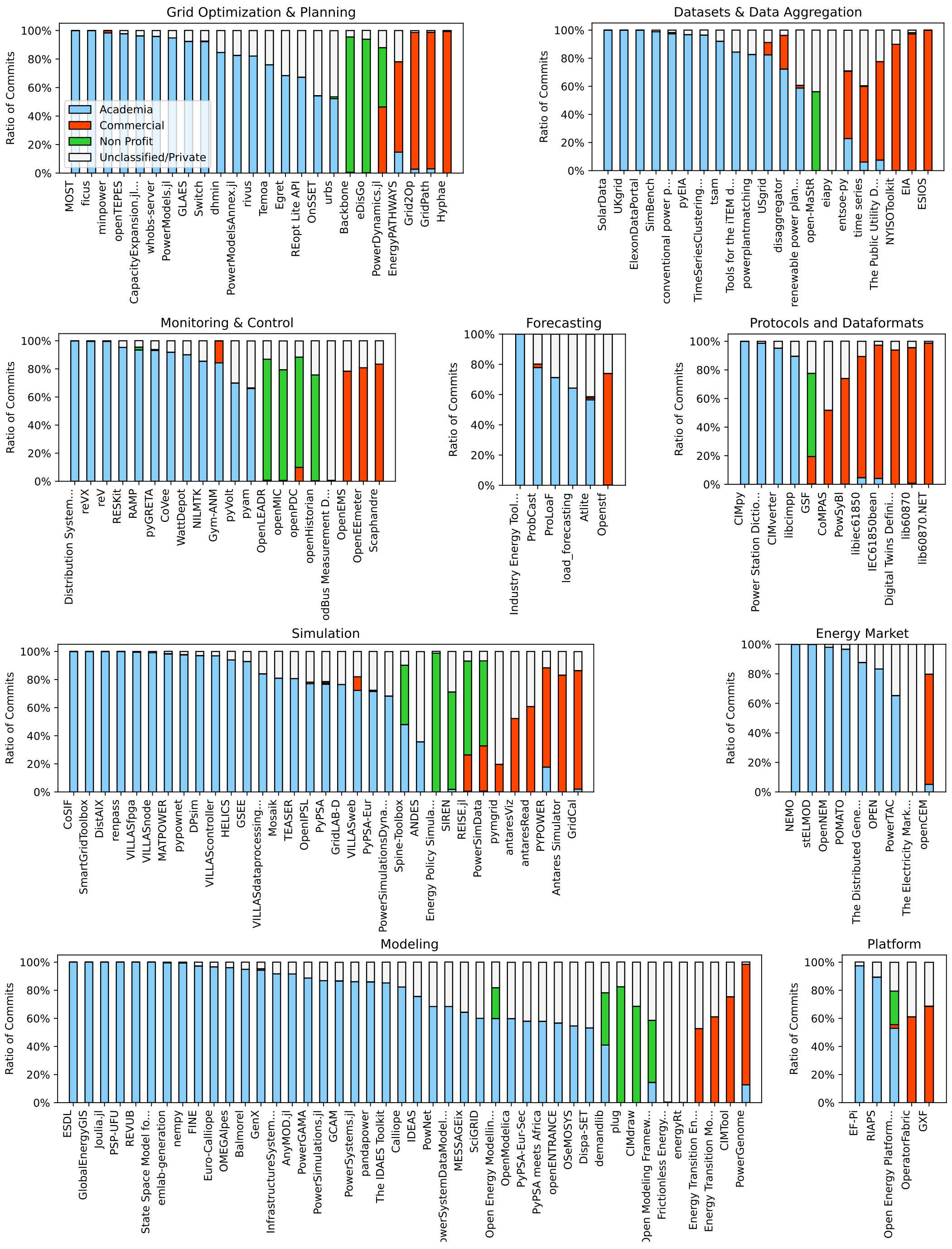

Figure 1. Percentage of electrical grid project's commits by sector, determined by email address.

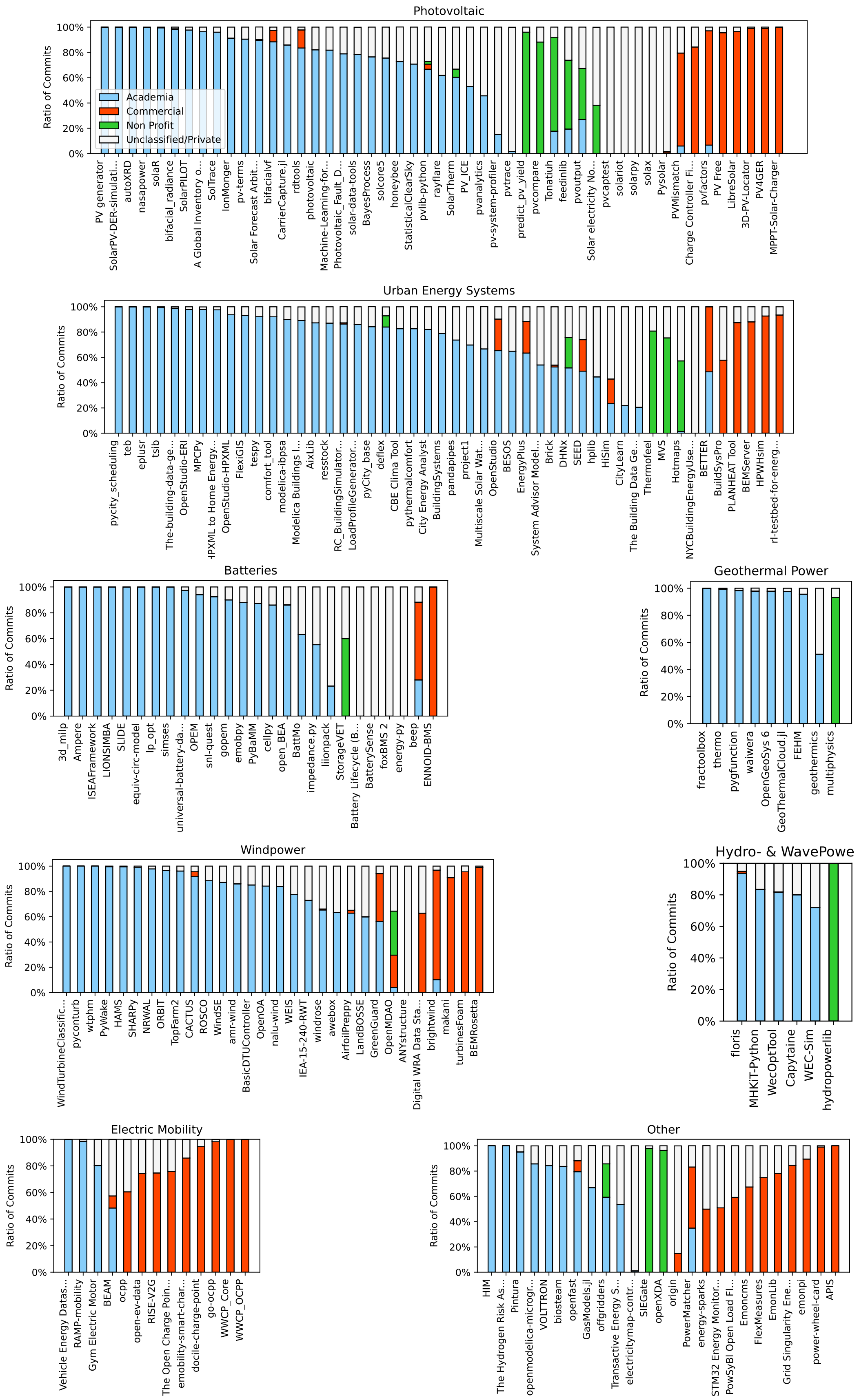

Figure 2. Percentage of other energy related project's commits by sector, determined by email address.

Other studies [36,39] consider more areas of the energy sector, but present the general State-of-the-Art in the respective disciplines. They do not focus on the state of open-source tools, and, hence, contain a mix of commercial and open-source projects in their collections.

Outside the energy sector, Androutsellis-Theotokis et al. [17] provide a survey of open-source software in general, including the history and evolution of open-source software, community aspects, open-source software licensing, and business models. The energy sector, however, faces unique challenges—such as security, reliability, and sustainability of the projects—in the adoption of open-source software principles, which limit the applicability of a sector-agnostic overview to the energy sector.

This is also highlighted by Pfenninger et al. [2], who conclude that energy research is behind other fields when it comes to openly published scientific results. They argue that making energy models, results, and the underlying data openly accessible improves the quality of related science, increases productivity because of collaboration, and leads to a more effective transfer of results into policy. However, the energy community is reluctant to do so, because of data privacy concerns, fear of unwanted exposure, discrediting authors, the time-consuming process of making results presentable, and a tradition of proprietary code and data.

The most comprehensive list of open-source projects we found was created by Protontypes [41]. With sustainability in general as their scope, they also covered sectors not relevant in this paper. In the categories we consider relevant from a grid perspective (Photovoltaics and Solar Energy, Wind Energy, Energy Modeling and Optimization, Energy Monitoring and Control, Energy Distribution and Grids, and Battery), we counted 264 projects in their data set. Protontypes analyzed the report, in terms of popularity, age, growth, ranking, programming languages, licenses, community, and users and usage. Most of these metrics were taken directly from or derived from GitHub metadata, which excluded projects hosted on other platforms from the analysis. Their analysis covered more sectors than the energy sector, so sector-specific findings cannot be derived from their report. Similar to our approach, described in Section 3.2, they also matched projects to sectors, but at the project level, not the contribution level. This allowed us to examine collaboration between different sectors within projects. Their report contains many valuable metrics, particularly for measuring individual projects; however, in this paper we focused on the energy sector, we analyzed more projects, and we looked more deeply at the composition of the community.

In summary, multiple collections with various focuses exist in the scope of open-source software for the energy sector. However, all the aforementioned catalogs contain some gaps, and we could not find all the information required in the existing data sets. Consequently, we built upon the presented work, by combining and extending the data sources, creating a more complete picture for our analysis than was previously available.

3.2. Community Composition

As discussed in Section 2.4, it is often questioned whether open-source software is compatible with commercial interests. To investigate this hypothesis, we analyzed the email addresses associated with each change stored in the git history of the project (commit), to categorize the contributions into Academia, Commercial, or Non-Profit, assuming that this data would give a strong indication of the affiliation of the author. Academia contained contributions from universities and non-commercial research institutions. Companies such as grid operators or consultancies were classified as Commercial. The category Non-Profit was for non-profit institutions without a focus on research or intentions of making a profit. However, it should be noted that these institutions are often backed by companies: for example, Linux Foundation Energy, which has multiple grid operators as members.

Often, the e-mail address domain could be assigned unambiguously to a company, an academic, or non-profit institution, even though some authors used public e-mail providers or multiple e-mail addresses for their contributions, making classification more challenging. To overcome this, we tried to research and classify the person and biography behind the private mail addresses of the most active contributors manually, to find affiliations with one of the proposed categories.

We did not include a category for projects that were mainly driven by private contributions, because we were not able to separate these contributions from the other categories, only by means of the commit author data. However, we found that only a minor portion of the projects seemed to be driven by hobbyists, which was likely a result of the scope of our selection.

As we relied on git metadata, we could only perform the analyses for the projects using git as a version control system, which corresponded to 97% of the projects. Furthermore, some projects were converted into a git repository at some point without preserving the entire contribution history. Thus, our analysis ignored contributions prior to this point. In total, we classified 753 email domains and individual addresses. An overview of the classification is given in Figures 1 and 2.

The classification of the single commits enabled us to sort entire projects into the aforementioned categories, if more than 50% of all commits could be matched to a single category. If a project was classified, but contained more than 5% of the contributions from another category, it was also additionally considered a Mixed project: we identified 32 such projects in the data set, which accounted for about 8% of the projects. Table 2 shows the project distribution according to this strategy.

Table 2. Distribution of the projects classified by major contribution affiliation.

Category	**Nr. Projects**	**Percentage**	**Share of Mixed Projects**
Academia	244	64.0%	6.6%
Commercial	71	18.6%	15.5%
Non-profit	26	6.8%	23.1%
Unclassified & Private	40	10.5%	-
Total	381	100%	8.3%

Our analysis showed that the majority of the open-source software projects in the energy sector were driven by academia. Approximately 18% of the projects were, to a significant degree, commercially driven. Academia being the largest contributor was somewhat expected, as open-sourcing code is often seen as part of the dissemination and publication of research work. The number of projects classified as commercial disproves the thesis that open source cannot be combined with commercial interests. We found that large companies, as well as small businesses and freelancers, were represented in the data for almost all the categories of software. In the Electric Mobility and Protocols and Dataformats categories, more than half of the projects were commercially driven.

Mixed contributions are generally desirable, to foster collaboration and transfer of scientific results to commercial products. With about 8% of the contributions coming from different sectors, this aspect was not negligible, but still the vast majority of projects kept development in their respective category. The share of mixed projects was significantly higher for non-profit and commercial projects than for academia. There are multiple possible explanations for this observation: for example, academia plays the role of a test field for innovative ideas and approaches, and only a smaller subset of the projects result in benefits for real applications and attract the interest of the other sectors. Another hypothesis is that there is less incentive in academia for collaboration, and that open sourcing is mainly done for visibility and prestige reasons.

3.3. License Distribution

In Section 2.1, we discussed the importance of the licenses in open-source projects. In our data set, we collected the licenses for each project, allowing us to analyze the licenses used in the energy sector. Figure 3 shows the distribution of the five most-used licenses of the cataloged projects: the permissive MIT, Apache, and BSD licenses, the copyleft GNU GPL family, and the Mozilla public licenses. Some projects used less common licenses or multiple licenses to grant different rights to various parts of the software. The category Other/Multiple combines both these cases, but makes up less than 5% of all licenses used in the catalog.

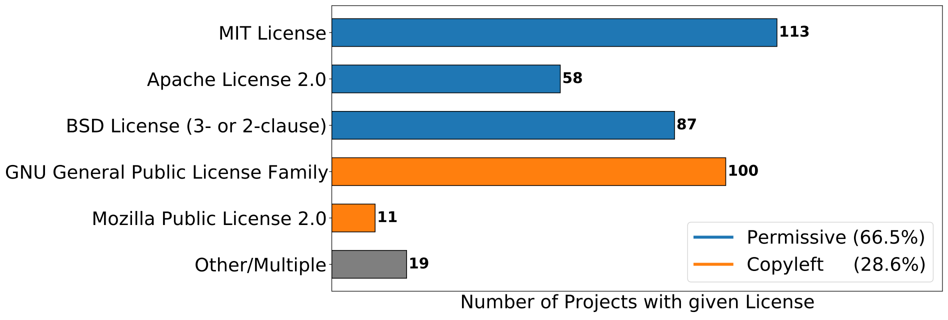

Figure 3. Most common licenses of projects in the catalog. Licenses used in less than 5% of the projects and projects with two or more licenses are categorized as *Other/Multiple*.

The gathered data shows that MIT license was the most-used license in the energy sector, closely followed by the GNU General Public License Family and the BSD license. Accumulated, 77.3% of the projects in our catalog used one of these top three licenses. Of all the analyzed licenses, 66.5% were permissive and 28.6% were copyleft. The remaining 4.9% included projects with multiple licenses or licenses that could not be clearly categorized into permissive and copyleft.

When comparing the distribution of licenses with the recommended licenses of the Open Source Initiative [12] (cf. Table 1), it is notable that most projects in our catalog used one of the recommended licenses that are already common in other sectors. This is interesting, as in the Protontypes analysis across more sectors, a five-times-higher proportion of projects were classified as custom. Hence, rather than causing further license proliferation, the energy community is supporting license compatibility between projects, allowing users and developers to extend or combine existing software into larger projects without the risk of legal issues due to non-compatible license combinations. As permissive licenses impose less restrictive rules regarding the usage and distribution of the licensed software, such projects tend to be easier to commercialize or find their way into an industrial context. Thus, the higher use of permissive licenses indicates a trend towards more commercial-friendly licenses in the energy community.

3.4. Common Programming Languages

Next, we analyzed the commonness of programming languages. As we were not interested in the languages for side tasks in a project, we only considered the main languages of each project, as determined by the language used for at least 80% of the lines of code in the project repository. Figure 4 depicts the most common programming languages among the projects.

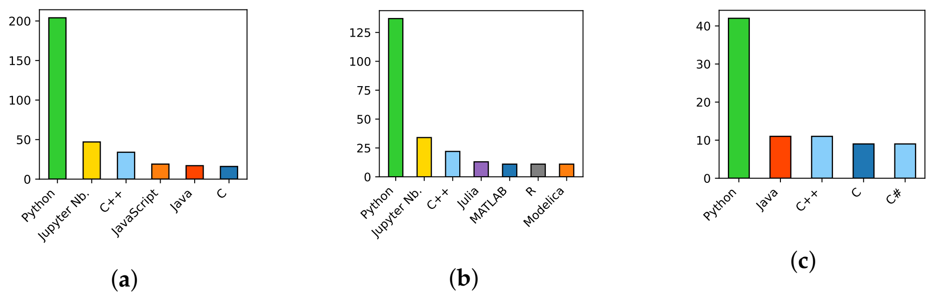

Figure 4. The most popular languages used in energy-related open-source projects (only the most common languages are shown): (**a**) all projects; (**b**) academically dominated projects; (**c**) commercially or non-profit-dominated projects.

This analysis shows that the Python programming language dominates the field of open-source software in the energy sector, with 204 of the projects (53%) using it as a main language. This was true for the academic classified projects (54% of the projects), as well as the commercial or non-profit ones (43%). Subsequently, differences in language popularity were shown for academically and for non-profit/commercially dominated projects. The languages Julia and R, which are connoted as academic languages, were among the most popular academic languages. In the commercial and non-profit sectors, Java and C# were more common—both languages that are commonly referred to as more popular in the enterprise domain.

When comparing these results to the trend of programming language popularity over the last few years [42,43], it is not surprising that Python led in this ranking. However, the ratio was significantly higher than what the popularity indices suggest. Notably, there was a prevalence of a single programming language, while domain-specific languages played a minor role. The Protontypes report had a similar finding when analyzing multiple domains, including the energy sector. In our larger data set, focusing only on the energy sector, the dominance of Python even increased, whereas R was more seldom. This could be beneficial for the future development of the open source landscape in the energy sector, as it enhances the interoperability of existing tools and ensures that upcoming generations of programmers are more likely to inherit the required skill sets for further development. Therefore, the use of such a language in a project increases the number of potential contributors.

3.5. Project Activity

Finally, we were interested in the activity and longevity of projects, so we evaluated the number of active projects per quarter of the year. We considered projects as active if there was at least one commit in the git history during the given quarter. The absolute number of commits per project was not considered, as it highly depended on the project's individual development workflow.

Figure 5 shows the number of active projects per given quarter, together with the overall number of projects. Additionally, the share of active projects as the quotient of the number of active projects with respect to the total number of projects is graphed.

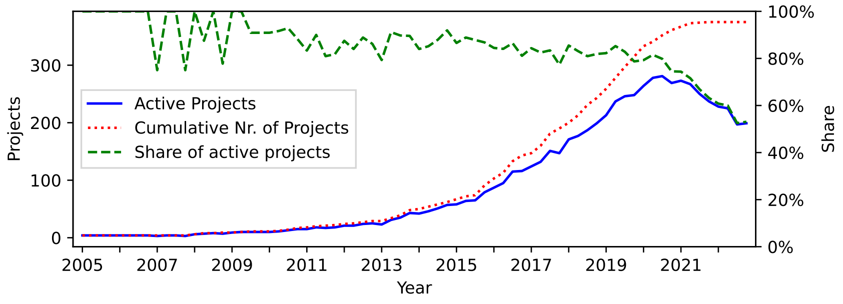

Figure 5. Number of projects with activity in the respective quarter, and total number of projects.

Until 2020, a steady rise in the number of projects and a high percentage of active projects can be observed. Starting in 2020, the rate of new projects declines, a trend that is also visible, but not explained, in the analysis of Protontypes. An explanation could be that recent new projects were not popular enough from the outset to be noticed by us, or the sources we conducted to build the data set. Another reason may have been that open-source software can be developed internally for quite some time before making it public. Earlier projects following this pattern were included in our data, but ongoing efforts were not.

It is also likely that this curve was partly a result of a survivorship bias in our data. Older projects which are not maintained anymore were less likely to be part of our catalog, whereas newer projects with a shorter lifecycle were more likely to be included.

This part of the data will be more relevant in future analysis, as the addition of new projects might overcome the current bias.

4. Conclusions

This paper provides an overview of the various aspects of open-source software, with a focus on the energy sector and good practices for open-source software projects. We discuss how, by embracing open-source software, the energy sector can profit from sharing development costs, fostering innovation, and growing the development interest, while ensuring software quality and security. When adopted at scale, open-source software has the potential to produce products that become de facto standards in the industry.

By discussing open-source software licenses and their implications for the redistribution and modification of software, we support developers creating an open-source strategy. In the energy sector, open source can increase the speed and innovation for interfaces, and thus the interoperability. Furthermore, we present different forms of open source communities and their governance, to understand the role of an active community behind an open-source software project. Presented business models for open-source software that are already successful in other sectors show that open-source licenses do not prevent the commercialization of software.

Our analysis of 388 projects shows a growing momentum for open-source software in the energy sector in recent years. We observed that most of the projects were compliant with good practices for licenses, and that Python was by far the most relevant programming language. However, collaboration between industry and academia should be fostered, as the majority of projects are still driven by academia. A more in-depth analysis of the community compositions revealed little overlap between communities of industry-driven projects and those of projects originating from academia. Therefore, we suggest continuing and strengthening existing incentives for open-source activities, while motivating industry to move towards open source, e.g., by showcasing success stories.

Author Contributions: Conceptualization, J.B., N.E., J.K. and F.W.; software, J.K. and F.W.; validation, J.K., F.W., J.B. and N.E.; formal analysis, J.K. and F.W.; investigation, J.B., N.E., F.W. and J.K.; data curation, J.K. and F.W.; writing—original draft preparation, J.B., N.E., J.K. and F.W.; writing—review and editing, J.K., N.E., F.W. and A.M.; visualization, J.K. and F.W.; supervision, A.M.; funding acquisition, A.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by the European Union's Horizon 2020 research and innovation program, under grant agreement No. 864300 (Platone) and No. 957246 (IoT-NGIN).

Data Availability Statement: Publicly available data sets were analyzed in this study. The data can be found here: https://git.rwth-aachen.de/acs/public/publications/2023-oss-in-energy-data/.

Acknowledgments: We would like to thank Markus Mirz for his inspiration for this topic.

Conflicts of Interest: The authors are involved in several open-source software projects themselves. The funders had no role in the design of the study, in the collection, analyses, or interpretation of data, in the writing of the manuscript, or in the decision to publish the results.

References

1. How to Build Open Source Competency in Your Company. Technical Report, Linux Foundation. Available online: https://web.archive.org/web/20211117154050/https://www.linuxfoundation.org/wp-content/uploads/Competency_Whitepaper_2015.pdf (accessed on 17 November 2021).
2. Pfenninger, S.; DeCarolis, J.; Hirth, L.; Quoilin, S.; Staffell, I. The importance of open data and software: Is energy research lagging behind? *Energy Policy* **2017**, *101*, 211–215. [CrossRef]
3. Digitalisation of Energy Action Plan: Linux Foundation Energy Response to European Commission Public Consultation. Available online: https://web.archive.org/web/20220129190425/https://www.lfenergy.org/wp-content/uploads/sites/67/2022/01/Linux-Foundation-Energy_EU_Consultation_Jan_2022.pdf (accessed on 19 January 2022).
4. Markard, J. The next phase of the energy transition and its implications for research and policy. *Nat. Energy* **2018**, *3*, 628–633. [CrossRef]
5. IRENA. *Global Energy Transformation: A Roadmap to 2050 (2019 edition)* ; International Renewable Energy Agency: Abu Dhabi, United Arab Emirates, 2019; ISBN 978-92-9260-121-8.
6. Gielen, D.; Boshell, F.; Saygin, D.; Bazilian, M.D.; Wagner, N.; Gorini, R. The role of renewable energy in the global energy transformation. *Energy Strategy Rev.* **2019**, *24*, 38–50. [CrossRef]
7. Manditereza, P.T.; Bansal, R. Renewable distributed generation: The hidden challenges—A review from the protection perspective. *Renew. Sustain. Energy Rev.* **2016**, *58*, 1457–1465. [CrossRef]
8. Faia, R.; Soares, J.; Pinto, T.; Lezama, F.; Vale, Z.; Corchado, J.M. Optimal Model for Local Energy Community Scheduling Considering Peer to Peer Electricity Transactions. *IEEE Access* **2021**, *9*, 12420–12430. [CrossRef]
9. IEA—Digitalization and Energy Working Group. *Digitalization and Energy*; Technical Report; IEA: Paris, France, 2017.
10. Linux Foundation Energy. Available online: https://web.archive.org/web/20211117162236/https://www.lfenergy.org/ (accessed on 17 November 2021).
11. DiBona, C.; Ockman, S.; Stone, M. *Open Sources: Voices from the Open Source Revolution*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 1999; ISBN 1-56592-582-3.
12. Opensource.org. Available online: https://web.archive.org/web/20211205004420/https://opensource.org/ (accessed on 5 December 2021).
13. Free Software Foundation Europe. Available online: https://web.archive.org/web/20211128194329/https://fsfe.org/ (accessed on 28 November 2021).
14. Fogel, K. *Producing Open Source Software, How to Run a Successful Free Software Project*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2005; ISBN 978-0596007591.
15. Various Licenses and Comments about Them—GNU Project—Free Software Foundation. Available online: https://web.archive.org/web/20210820084648/https://www.gnu.org/licenses/license-list.en.html (accessed on 20 August 2021).
16. Licenses & Standards—Open Souce Initiative. Available online: https://web.archive.org/web/20210730190629/https://opensource.org/licenses (accessed on 30 July 2021).
17. Androutsellis-Theotokis, S.; Spinellis, D.; Kechagia, M.; Gousios, G. Open Source Software: A Survey from 10,000 Feet. In *Foundations and Trends in Technology, Information and Operations Management*; Alet Heezemans: St. Louis, MO, USA, 2011; Volume 4, pp. 187–347. [CrossRef]
18. Goldman, R.; Gabriel, R.P. *Innovation Happens Elsewhere—Open Source as Business Strategy*; Morgan Kaufmann Publishers: Cambridge, MA, USA, 2005; ISBN 1558608893.
19. West, J.; O'Mahony, S. Contrasting Community Building in Sponsored and Community Founded Open Source Projects. In Proceedings of the 38th Annual Hawaii International Conference on System Sciences, Big Island, HI, USA, 6 January 2005; IEEE: Piscataway, NJ, USA, 2005; p. 196c. [CrossRef]
20. Izquierdo, J.L.C.; Cabot, J. The role of foundations in open source projects. In Proceedings of the ICSE-SEIS '18: 40th International Conference on Software Engineering: Software Engineering in Society, Gothenburg, Sweden, 27 May–3 June 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 3–12. [CrossRef]
21. FIWARE Foundation. Available online: https://web.archive.org/web/20211117162113/https://www.fiware.org/ (accessed on 17 November 2021).
22. openADR Alliance. Available online: https://web.archive.org/web/20211117161749/https://www.openadr.org/ (accessed on 17 November 2021).
23. Okoli, C.; Nguyen, A. Business Models for Free and Open Source Software. *SSRN Electron. J.* **2016**. [CrossRef]
24. COSSI: $100M+ Revenue Commercial Open-Source Software (COSS) Company Index. Available online: http://web.archive.org/web/20211125133406/https://docs.google.com/spreadsheets/d/17nKMpi_Dh5slCqzLSFBoWMxNvWiwt2R-t4e_l7LPLhU/edit (accessed on 25 November 2021).
25. Lewis, G.A.; Morris, E.; Simanta, S.; Wrage, L. Why standards are not enough to guarantee end-to-end interoperability. In Proceedings of the Seventh International Conference on Composition-Based Software Systems (ICCBSS 2008), Madrid, Spain, 25–29 February 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 164–173.
26. Kupzog, F.; Genest, O.; Ahmadifar, A.; Berthome, F.; Cupelli, M.; Kazmi, J.; Savic, M.; Monti, A. SGAM-based comparative study of interoperability challenges in European flexibility demonstrators: Methodology and results. In Proceedings of the 2018 IEEE 16th International Conference on Industrial Informatics (INDIN), Porto, Portugal, 18–20 July 2018; pp. 692–697. [CrossRef]

27. Wright, S.A.; Druta, D. Open source and standards: The role of open source in the dialogue between research and standardization. In Proceedings of the 2014 IEEE Globecom Workshops (GC Wkshps), Austin, TX, USA, 8–12 December 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 650–655.
28. Pau, M.; Mirz, M.; Dinkelbach, J.; Mckeever, P.; Ponci, F.; Monti, A. A Service Oriented Architecture for the Digitalization and Automation of Distribution Grids. *IEEE Access* **2022**, *10*, 37050–37063. [CrossRef]
29. Petitcolas, F. La cryptographie militaire. *J. Des Sci. Mil.* **1883**, *9*, 161–191.
30. Mercuri, R.T.; Neumann, P.G. Security by obscurity. *Commun. ACM* **2003**, *46*, 160. [CrossRef]
31. Ruffin, M.; Ebert, C. Using open source software in product development: A primer. *IEEE Softw.* **2004**, *21*, 82–86. [CrossRef]
32. Payne, C. On the security of open source software. *Inf. Syst. J.* **2002**, *12*, 61–78. [CrossRef]
33. Linux Foundation Security. Available online: https://web.archive.org/web/20211109120739/https://lfx.linuxfoundation.org/tools/security/ (accessed on 9 November 2021).
34. Bazilian, M.; Rice, A.; Rotich, J.; Howells, M.; DeCarolis, J.; Macmillan, S.; Brooks, C.; Bauer, F.; Liebreich, M. Open source software and crowdsourcing for energy analysis. *Energy Policy* **2012**, *49*, 149–153. [CrossRef]
35. Hecker, F. Setting up shop: The business of open-source software. *IEEE Softw.* **1999**, *16*, 45–51. [CrossRef]
36. Lam, H.L.; Klemeš, J.J.; Kravanja, Z.; Varbanov, P.S. Software tools overview: Process integration, modelling and optimisation for energy saving and pollution reduction. *Asia-Pac. J. Chem. Eng.* **2011**, *6*, 696–712. [CrossRef]
37. Groissböck, M. Are open source energy system optimization tools mature enough for serious use? *Renew. Sustain. Energy Rev.* **2019**, *102*, 234–248. [CrossRef]
38. Holmgren, W.F.; Hansen, C.W.; Stein, J.S.; Mikofski, M.A. Review of Open Source Tools for PV Modeling. In Proceedings of the 2018 IEEE 7th World Conference on Photovoltaic Energy Conversion (WCPEC) (A Joint Conference of 45th IEEE PVSC, 28th PVSEC 34th EU PVSEC), Waikoloa, HI, USA, 10–15 June 2018; pp. 2557–2560. [CrossRef]
39. Ringkjøb, H.K.; Haugan, P.M.; Solbrekke, I.M. A review of modelling tools for energy and electricity systems with large shares of variable renewables. *Renew. Sustain. Energy Rev.* **2018**, *96*, 440–459. [CrossRef]
40. Free Software Directory. Available online: https://web.archive.org/web/20211202055700/https://directory.fsf.org/wiki/Main_Page (accessed on 2 December 2021).
41. Augspurger, T.; Malliaraki, E.; Hopkins, J. *Open Source in Environmental Sustainability*. Available online: https://web.archive.org/web/20230609005547/https://report.opensustain.tech/chapters/index.html (accessed on 9 June 2023)
42. TIOBE. TIOBE Index for February 2022. Available online: https://web.archive.org/web/20220216010905/https://www.tiobe.com/tiobe-index/ (accessed on 16 February 2022).
43. PYPL PopularitY of Programming Language. Available online: https://web.archive.org/web/20220209101753/https://pypl.github.io/PYPL.html (accessed on 9 February 2022).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.