Networks of (anti-)trust

Implicit development environments in FOSS infrastructure communities and the future of public good(s) by private means

Literature Review for **IDE**

Author: Katharina Meyer
Project Collaborators: Elisa Lindinger & Julia Kloiber

All maintenance is politics (Marres and Lezaun 2011)
The need for maintenance (labor) overwhelms the need for freedom (action) (Kelty 2013) after Arendt)

Abstract

The aim of this literature review is to present the relationships between value-based self-organization and market-oriented organizational theory from multiple perspectives. Maintenance activities have a particular vanishing point in this regard: maintaining functionality, relationships, and dependencies in a network of behavioral regimes as well as social and technical standards. This network can also be described as "implicit development environments."

What starts off as a "technical" sphere becomes politicized when it comes time to allocate resources, if not sooner. This politicization in the form of non-material influences on technologies needs investigation with references from various disciplines to identify areas requiring further research.

Open-source development is a social process that produces information as a public good. Inputs in the form of labor and capital flow into this knowledge-intensive transformation, while, on the output side, information goods emerge which, as a special distinguishing feature, are neither excludable nor rivalrous goods[1].

This suggests a bridge between anti-trust law and networks of trust[2] or production communities. The "market" for open source and its field rules essentially unfolds between these two poles and negotiation mechanisms.

[1] Excludability
[2] F.i., Linux Kernel has been described as "*a lieutenant system built around a chain of trust*" (see 2. How the development process works — The Linux Kernel documentation)

Infrastructure[3] as used here is most effectively described with the metaphor of a stack, the connection to protocols and standards, and a consideration of the parallel development of the internet and open-source principles (one as a prerequisite for the other, as shown in Weber 2000; Benkler 2002, among others.

The review then concludes by offering reasons why the promotion of open infrastructures should be considered public services and thus fall into the remit of the non-profit (the so-called third) sector and the state.

Methodology

A keyword search was used to find articles in journals and monographs in the fields of (technical) sociology, organizational sciences and management theory, internet studies, economics, as well as cultural and media studies which deal with the interplay among so-called peer production, the "social shaping of technology," and values-oriented group dynamic processes in software development.

Germonprez et al. (2018) sensibly divides existing literature into the following categories:

1. *Conceptual*
2. *Performance metrics*
3. *Legal & regulatory*
4. *Production*
5. *Diffusion*
6. *Beyond software*

This review is likewise structured according to these basic areas, concluding with findings that might be used for further original quantitative or qualitative research.

Limitation to the tension between anti-trust and networks of trust

This review situates the extent to which the commodification of open source being based on values affects the current failure of certain infrastructure projects to find a market.

Based on the review of existing literature, a proposal is made for future research to focus even more on the role of the various resource management and governance regimes.

Below, the individual fields of observation that were conspicuous during the clustering are discussed, albeit in a brief and paraphrasing manner; a detailed bibliography and footnotes indicate sources for further information on structurally important questions.

[3] See Digital Infrastructure

Preliminary considerations: commons, markets, (doubly codified) values

Open source has become a well-developed field of research, with many contributions into the concept itself, in terms of performance indicators, from a legal and regulatory point of view, the development of production processes, distribution mechanisms, and even how the concept might be applied beyond software development (See Aksulu and Wade 2010).
Despite sufficiently described collective-action problems and just as carefully documented socio-technical hopes for commons (See Benkler 2016, et al.), there are still misallocations of (monetary and non-monetary) resources in the OSS ecosystem that the market alone apparently cannot balance.[4]

Such conflicting resource management regimes are inherent given of the origins of open-source software and its integration into the growth strategies of the ICT companies that dominate the market.

One of the most extensive summaries of these developments is made by Hill (2013) who convincingly demonstrates the historical influence of Eric Raymond, who, in the formative years of FOSS, framed peer production as "superior" to previous models for software production (Raymond 1999).

As a result, an illusion has been perpetuated, even "inscribed" in the analysis of open-source projects where any problems in peer production should, in theory ((See Ostrom 1990; et al., be balanced out with "collective action," although this does not, in large part, correspond to the realities of production. Hill (2013) writes:

> "*Both members of the FLOSS community (e.g., (Hill 2005)) and (other) sociologists (e.g., (Lin 2005)) have tried to re-frame Raymond's work as a Utopian ideal rather than a representation of empirical reality. Large cross-sectional analyses of FLOSS projects by (Krishnamurthy 2005)), (Healy and Schussman 2003)), and (Scaria 2013) have presented empirical evidence that calls into the question claims of FLOSS collaborativeness and has shown that participation in FLOSS projects is "skewed": "the median number contributor to a project is one" (Healy and Schussman 2003).*

Nevertheless, most of the literature on peer production in the software industry is based on the few successful examples of collective resource management, often those involving companies.

[4] cf. Heartbleed - Bug (https://heartbleed.com): In OpenSSL, a faulty commit from one (of the few) project maintainers was integrated in the source code by the end of 2011, which subsequently presented an attack surface. The vulnerability was only discovered and fixed in spring 2014 - after 27 months. The case is so interesting because only a few people maintained OpenSSL, and most of them did so as volunteers. At the same time, OpenSSL was built into many other software products, including those of large companies. For example, Google's Android system was also affected by the vulnerability. The scaling only took place in the usage, but not to the same extent in the human and attention resources for maintenance.

Based on these findings, it can therefore be abstracted that most of the systematic considerations that have historically been available about open-source software have no meaningfulness for the current analysis of digital infrastructure and that new research material will have to be generated here in the future.

One can justifiably state that the "differentiation expectations of digital modernity" have not materialized in FOSS (Schrape 2016): The narrative of open-source development as a "revolutionary production model" based on voluntary and self-directed collaboration that is superior to established socio-economic coordination methods such as market and hierarchy and forces established companies to make adjustments should be considered an exaggeration (See (Dickel and Schrape 2015). Following Andreesen[5], it can therefore be said that software has indeed incorporated the world that created it. FOSS as a production mechanism both won and lost the initial conflicts about its cultural history (O'Neil et al. 2020).

Another observation on implicit development environments that is relevant to the research question: although existing research shows that a lack of participation in software projects does not necessarily equate to failure, it can lead to a failure in scaling if the freerider economy[6] is not balanced with investment.

If one is therefore interested in a reintegration of the market dynamics into a public interest framework, which was also the basis for including a consideration of FOSS infrastructure in the funding portfolios of the Ford Foundation[7], it is then necessary to consider the social functions and hopes that are linked to open-source software, since these do influence resource allocation.

Licenses are a socio-technical unique selling point for open-source development, as they have to date been the infrastructural units that regulate the dynamics between collaborative production and market interests and have been the source of FOSS' success.

The production conditions for OS software have not fundamentally changed since the 1990s, but the number of their "dependencies" has: open source has since become "the most successful software tactic of the present day."[8]

Open-source software and the commercial IT market are closely interwoven. Open-source architectures have significant market shares[9] in the area of basic ICT infrastructure, with companies combining free and proprietary components. For example, considers its integration in Google and Facebook's product suites to smart TVs and the vast majority of all cell phones.

[5] Why Software Is Eating the World
[6] Free-rider problem
[7] Every day, we rely on digital infrastructure built by volunteers. What happens when it fails?
[8] Open-Source-Software - Die neue Offenheit - Digital - SZ.de
[9] Warum quelloffene Software die Welt regiert - Digital - SZ.de

It was Red Hat and Suse, together with a community of developers (and meanwhile many major IT players), that laid the foundation for the modern cloud. Open source also plays a key role in many other emerging technologies.

The last fifteen years have consisted of the "*corporatization of Open Source*" (Hogan 2009 in Planet MySQL). The wide array of licenses apparently extends the strategic options, especially for commercial stakeholders (Lerner and Schankerman 2010)).

Connection with anti-trust law and market shares

This led to an increased need for regulation. Examples include Google, which dominates the Android system via the Open Handset Alliance[10]. The interlinking with proprietary components led to an antitrust review in 2015[11].

There are currently a number of pending lawsuits seeking sanctions against companies that have achieved their market dominance through the intelligent integration of FOSS into their growth strategy (this applies in particular to Amazon Web Services, AWS for short[12]).
As Schrape (2016) writes,

> *"In the worst case, narratives of openness and the digital reconfiguration of society mask development trends that go against the ideal of a more open and democratic economic world, for example a creeping erosion of 'the historically evolved system of regulation of work' (Boes et al. 2014, p. 71), the restriction of fundamental consumer rights through the terms and conditions of many online services, and a global concentration of providers in the field of communication technology infrastructures that is the first of its kind in the history of media."*

Originally, (open) software development and the industry that surrounds it got its boost from antitrust proceedings initiated in 1969 against IBM, which was accused of trying to force potential competitors out of the market by offering hardware, software, and services (Neubert 2015).
This was followed by the historic decision in favor of "unbundling" and the distribution of permissive licenses, which initiated an opening to the market, but were not entirely "toothless:"
from 2001, violations of the GPL were the subject of several legal proceedings in Europe and North America which forced companies like Skype and D-Link to remove questionable software elements in their proprietary program architectures or otherwise make their source code freely available[13].

[10] Geboren aus dem iPhone-Schock: Zehn Jahre Android-Allianz
[11] Europe opens antitrust investigation into Android
[12] https://judiciary.house.gov/uploadedfiles/competition_in_digital_markets.pdf
[13] Open source license litigation

Nevertheless, there is currently a clear trend towards taking advantage of developers while the principles of open innovation are allowing companies in particular to generate additional revenue. In order to explore the normative and (behavioral) economic basis for this, the history of the concept of free software production should be considered in order to shed light on the potential for conflict in its management.

Brief history of open source

In his study, Schrape (2016) describes four phases of FOSS development: collective invention (1950s), commodification and the genesis of a subculture (from the 1960s), institutionalization (since the 1980s), growth & diversification (starting in 1998).

1998 represented a turning point in the approach to building software commons[14]:

"Open Source" replaced "free software" with the emergence of the Open Source Initiative and became "a matter of faith" that brought about both material and ideological differences (Eghbal 2016) and was actively constructed around infrastructure, standards, and competitive, commercial power. As shown, non-excludability is a prerequisite for open source, but is also problematic, perhaps its most problematic aspect.

In connection with FOSS, OSI initially promoted the narrative of economic competitiveness as core and aligned it to the mainstream of the commercial and corporate world.
What counted was that the software was "technically excellent" due to its openness and verifiability, no longer its "moral superiority."[15]

> *"Free Software had done us damage over the years because of the strong association of the term 'free software' with hostility to intellectual property rights, communism and other ideas hardly likely to endear themselves to an MIS manager"* (Raymond 1999)

The increasing importance (or "criticality") of software infrastructure for companies and research corresponds to a further politicization: the hybridization of the commercial logic of technology companies and the often more communal logic of software community projects require rhetorical legitimation and sometimes lead to an undersupply of monetary and non-monetary resources (O'Neil et al. 2020).However, resources are fundamental to resilience.

After all, an important and widely used argument (see above) for open source is the forecast increase in security through regular, transparent code reviews and hardening against attacks in the digital space. However, this security is not an automatic mechanism, but is instead based on a cleanly monitored and maintained open-source ecosystem, in which sufficient resources (and

[14] Digital commons (economics)
[15] opposite to Stallman etc.

knowledge) not only flow back into technical maintenance and scaling, but also into the host organizations and maintainers (See Benthall et al. 2016), among others.

Due to measurable failure phenomena such as the Heartbleed bug, some open stack components are now well secured through industry alliances, such as the Core Infrastructure Initiative[16] or the Open Source Security Foundation[17]. For other software modules and dependencies, there continue to be clear cases of *code debt*, as the sufficiently described collective action and free rider tendencies in open source sometimes advanced projects to the critical software infrastructure simply due to their availability, while their maintenance could not be scaled equally (Eghbal 2016).
Another risk arises due to insufficient analysis of systems and falling entry thresholds for web development (for example, unconsolidated code bases, above all the quality problems in the Javascript community with the node libraries, see Pano et al. 2016)

> *"Projects can be stuck between professional & egalitarian ideals - this potentially leads to public coordination and maintenance failures."* (Germonprez et al. 2018)

The development and maintenance of digital infrastructure is still largely the work of highly qualified contributors, some of whom are "one-person shops" that can become fundamental for the introduction and successful operation of hundreds of new projects.

FOSS software and its ecosystem face the challenge of collective action that arises from the tensions between the socio-technical past and the integrated, corporate future.

Even if individual open-source projects serve a specific technical need and remain relatively small in terms of the number of participants and contributions, upon adoption of their products they become part of a complex upstream and downstream network of continuously integrated open-source technologies. *"Maintainers must understand the significance of the supply chains their projects become part of"* (Germonprez).

Coleman (2013) describes the contribution of code to open-source projects as "*individual agency combined with a convivial but competitive collective experience*."

The experience of competitive dynamics comes from the fact that even the commons are not free from the forces of the market. Kelty (2013) goes on to explain how this fact is directly related to the formation of governance structures, which "*over the years have transformed some free software projects into open-source projects*."

As Germonprez et al. (2018) suggest, it seems important to consider that "*open source is in fact a complex collection of interrelated phenomena [...] and not a single thing at all."*

[16] Core Infrastructure Initiative: Home
[17] Open Source Security Foundation: Home

Infrastructure - subcultures

With the ubiquitous networking of the world, open software basic technologies and the cooperation for their development are becoming increasingly important, as described. They form the prerequisites for innovation and the emergence of more agile software and subordinate industries.

They are mainly implemented at the infrastructure level. On the hardware side, infrastructure often describes massive network technologies that connect devices with one another. In order for the server and end device to use the solution stack[18] to render the individual internet protocol layers from human to machine and back again and thus make smooth communication possible, something else is also necessary: software infrastructure.

This can include such things as software packages[19] or standardized implementations[20] of protocols used by developers to write application software and operate it. Code modules are exchanged, expanded, and reused via repositories[21] and package management[22], because open-source software is essentially modular and is often useful in a wide range of applications.

Infrastructure technologies can be developed from their historically situated assemblage[23] by using archives (such as mailing lists) to artificially create the “state of crisis” that led to their conception and dissemination (Straube 2016).
This shows contingent design possibilities and the resulting technical affordances[24] which still define the basic formats, mechanisms, and user interfaces of a broad spectrum of today’s applications.

Weber (2000) points out that infrastructure projects also differ from applications in particular in that they over-exemplify *user innovation* (per Von Hippel 2005) with integration decisions first being made by developers. Infrastructure products are thus rated according to their technological superiority and technical properties, specific value scales within a meritocratic system (on the challenges of meritocracy, see Dunbar-Hester 2020) that inherently does not produce for market share, but for the solution of existing problems.

[18] Solution stack
[19] Software suite
[20] Software implementation
[21] Software repository
[22] Package manager
[23] Assemblage (philosophy)
[24] The concept of "affordance" stems from cognitive psychology, where it has been used to indicate uses and functions that are materially inscribed in physical objects or fields of action. Thus, the affordances of an object should describe its phenomenological properties, signal the possible functions of the object, and inform us about potentials and limits of its usability.

The coordination structures are, however, characterized by being interwoven with corporate contexts; they have emerged through growth and tiered management structures that serve quality assurance. Quality standards also arise when users "vote with their feet." While, fundamentally, there are dynamics in open source that should prevent forks[25] as much as possible, from an institutional level these can be a valid means of achieving the purposes pursued by companies seeking to protect their supply chain.

Computers and software are structurally open and protean machines[26] that are ontologically indeterminate. Only through their "communities of practice" do they become media[27].

Standards

Since the pioneering work on the genesis of open source referenced above, the motivational requirements for developers and FOSS[28]' penetration of the technological sphere have changed.

As shown, high market shares force companies to develop open source and market strategies. The goal is often to increase the compatibility and interoperability of their own products[29].
The market and the growth of the IT economy are directly influenced by standards.

Standards also form an important differentiator for the definition of software infrastructure and particularly affect three classes of actors (as described in the qualitative section of IDE[30]):

- Standard creators: people and organizations who decide upon and write IT standards.
- Standard implementers: people who implement these standards in their software and create digital tools and services around them.
- *Service providers and maintainers: people and organizations who run these tools and services, and can directly observe how these standards affect users.*

Niels ten Oevers' recently published dissertation "Wired Norms", especially Chapter 3, rigorously analyzes how, in the wake of the privatization of internet architecture in the 1990s and the advent of standardization committees, the interplay between the architectural principles of "permissionless" innovation and openness undermined the equality among internet users and reconfigured the affordances of internet architecture, which subsequently made it easier to prioritize company interests over the interests of end users (ten Oever et al. 2020).

---

[25] Fork (software development)
[26] protean
[27] https://www.mediacoop.uni-siegen.de/en/projects/a01
[28] Free and open-source software
[29] Techcrunch: How-open-source-software-took-over-the-world/, https://www.forbes.com/sites/janakirammsv/2017/07/09/how-google-turned-open-source-into-a-key-differentiator-for-its-cloud-platform/?sh=4970a5a7646f & A history of IBM's open-source involvement and strategy
[30] https://recommendations.implicit-development.org/project-details/

As in previous work, the analysis of the "entanglement" of values (or their lack of them) in technical standards, the relationship and the power equilibrium (or lack thereof) among a multitude of public (e.g. states) and private actors (e.g. industry, consumers), informal legislation and multi-stakeholder governance mechanisms inform a theory of political economy for FOSS.

In general, standards are developed by various actors (such as network operators, device manufacturers, content and service providers, and software developers) in a voluntary, consensus-oriented process in order to facilitate interoperability among products.

In some cases, the broader open-source community actively invests in these bodies to prevent business-critical technology from being dominated by a single or just a few companies. The number of independent companies that contribute to a particular software product is an important measure of community health in practice.

Studies like that of Thumm (2019) provide an insight into how the role of OSS projects in innovation processes has historically changed and also why large companies have decided to invest in and steer development in this area. With the three archetypes of technical standardization: "standards as input in OSS," "OSS as input in standardization," and "parallel development," the toolset of market penetration has shifted. Where standards were once considered necessary to achieve interoperability, the OSS development process now offers an alternative way to achieve this interoperability.

Software-oriented implementation projects rarely adopt specifications that are then implemented by the FOSS projects as reference implementations. They are usually not the places to develop new and innovative technologies. More often than not, open-source communities develop new technologies which then become candidates for standardization.

Thumm et al. ((2019) systematically show that "*through the evolution of open-source governance towards standardized behavioural norms*," the open-source community continues to build up a common basis of freely available basic software technologies. This basis has been expanded through the granting of patent licenses and the increasing acceptance of open-source licenses which include patent licenses.

Among other things, if a provider has absolute market power, there is a risk of quasi-standards: "*because one arbiter fully controls the development of the OS, it can determine the technological specifications to which partners must abide*" (Spreeuwenberg and Poell 2012). See also Dolata (2014).

Apple could serve as an example of this. In contrast to Microsoft, it used the advantages of open source early on, a decision based on Apple's specific company history[31]. MacOS & IOS

[31] https://apple.stackexchange.com/a/401881

are based on the Darwin OS and kernel[32], and contain more than 200 other open-source components. The Darwin developer base therefore consists almost exclusively of Apple developers:

> *"In the beginning, Apple used to make Darwin available as a separate OS, including compiled binaries, installers, ISOs, etc. that you could install on Apple hardware. However, for many years now, Apple only provides a source code dump, every time a new release of macOS comes out. It isn't even possible to compile this source code, because it depends on Apple's internal build tools and build pipeline. There have been some projects trying to patch Darwin to compile it with publicly available tools, but those projects have all died from lack of interest."*

The so-called "browser wars"[33] are also relevant for understanding the connections among market power, standards, and open alternatives.

Due to the close relationships between SPOs and OSS communities described above, complementarities, possibilities for interaction, and differences between open source and standardization processes should also be considered in this literature review.

As Gamalielsson and Lundell (2017)) state, there are several studies that address organizational participation in open-source projects, but fail to address standards and their implementation.
Other studies examine organizational collaboration in certain open-source communities through approaches that include social network analysis (see Tamburri et al. 2019).

There is further research that focuses on organizational aspects, such as the different motivations of firms to participate in open-source projects, and the way firms engage in open-source projects under different governance models (e.g. aspects of community building in business-supported communities) and the increasing involvement of professional and commercial organizations in OSS. However, none of the studies on these issues explain how organizations actually get involved in specific cases.

The available studies also point to the finding that the few cases of close interaction between open-source software and standardization are mainly aimed at consortia that have a strict RF and more patent-intolerant licensing policy, i.e. W3C or OASIS. In contrast to formal SDOs that use the FRAND schem[34]e, these are better able to integrate input from OSS projects.

An important difference in more formal settings, i.e. between standardization bodies and open-source communities, is that the recognized status enables the SDOs to make and implement decisions with which a decisive minority may not agree.

[32] Darwin (operating system)
[33] Browser wars
[34] Reasonable and non-discriminatory licensing

In general, the actors do not have the option of "forking," i.e. making a copy of an open-source project and continuing its development separately. In OSS communities, however, relevance depends on the community's ability to reach consensus among a critical mass of stakeholders on both technical alignment and governance norms.

The politics of setting standards is typically analyzed in terms of the bargaining power wielded by corporations and national governments. The OSS community is not a company and is not represented by a state; nevertheless, individual representatives can be effective in standardization bodies.

This process repeats what we know about how epistemic communities[35] influence environmental bodies and NGOs' influence in human rights issues (see Dobusch u. Quack 2008, et al.).

In summary, it can be said about the current situation of open source as an innovation strategy that

- Corporate investments are changing the open-source ecosystem. Nowadays, open source often aggregates and promotes the interests of profit-oriented organizations and this commitment transforms the software components. Often this influence is implicit, not explicit.
- Power and the market are mediated in SPOs, among other things
- Corporate communalism (as a form of social computing[36]) is predominant (see Germonprez et al. 2018)

Communities

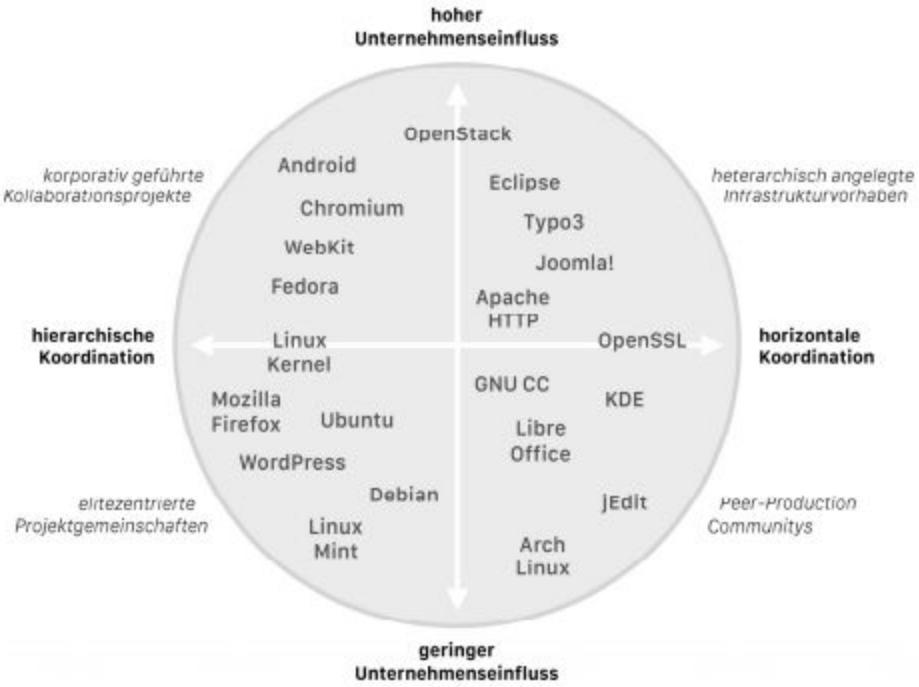
A clear result of studies like Lakhani and Wolf (2003) is that the F/OSS community is heterogeneous in terms of its motives for participation. These results are consistent with collective action research, in which group heterogeneity is seen as an important characteristic of successful movements (Marwell and Oliver 1993).

In order to be able to make a meaningful differentiation into subgroups, Weber (2000) suggests considering, in particular, these three dimensions in communities:

- Motivation of individuals
- Coordination mechanisms
- Complexity

[35] Epistemic community (international relations)
[36] Social computing

Schrape (2016) has identified four classes of projects associated with the various categories:



From corporate-led collaboration projects and elite-centered project communities through heterarchical infrastructure projects to egalitarian developer groups that most closely correspond to the idea of the original commons-based peer production.

According to the literature, a special area of tension exists between elite-centered communities and peer-production communities. Elite-centered project communities are characterized by organic growth, leadership circles, or benevolent dictators[37] working for the good of the communities in mind, since forking is always an available option for forming new communities instead.

Corporate collaboration, on the other hand, solves knowledge-sharing dilemmas[38] (market). This describes problem-centered collaboration on an individual level or as a representative of a company.

“Enthusiast projects” such as Arch[39], Dragora[40], and Parabola[41] therefore exist alongside company-driven projects and software from and for research and development. These specialist products are geared towards very specific stakeholder groups and, thanks to the group

[37] Benevolent dictator (for life)
[38] Knowledge-Sharing Dilemmas
[39] Arch Linux
[40] Dragora
[41] Parabola GNU/Linux-libre

homogeneity, were able to do without a pronounced social structure. When it comes to their boundaries and when taking governance decisions, however, they tend to be less open to expansion and expectations. But the strict cohesion keeps these projects comparatively stable.

OSI[42] drove and codified a cultural framework for open software development, as described, that places "excellence over zeal," and underscored the rationality component of technical decisions, which are at least partially driven by the need to maintain cooperation between communities and companies. This legitimized concerns about the "maintainability of the code," the cleanliness of interfaces, the clear and unambiguous modularity: *"Let the code decide."*

Karl Fogel and Mozilla published a leaflet with open-source project archetypes in 2018 (Fogel 2018), which schematically showed which use cases can address the "*benefits of open source*" that had been insinuated. These cases also include infrastructure-specific ones.

Knowledge and work practices in open software development

The process within a community from the idea to the code commit can be described schematically as follows: content creation, gatekeeping, quality filtering, integration, and delivery.
By analyzing the organizational structure, the decision-making processes, and the social order of a community as key elements of its governance, it is possible to compare communities in terms of their nature, even if they create unrelated products.

What distinguishes online peer production from many other peer production fields is the wealth of digital trace data. As a result, much previous research has taken a quantitative approach.
Often these studies are methodologically based in social network analysis, which uses software to analyze communities.
However, as Lindinger / Kloiber also point out in the main report, this methodology is limited by the infrastructure developer mindset that potentially inhibits which data points it can find.

> *"Even when their work is publicly known, they are (...) less visible (...). Since most of these projects value privacy and decentralization, they tend not to share their code on GitHub, but through other, decentralized channels such as GitLab, Bitbucket and Sourceforge. Most research has focused on GitHub because its centralized structure enables researchers to query data on large numbers of projects. However, this has led to projects present on other channels not being represented in studies into open digital infrastructure."*

Germonprez and the CHAOSS project have been working for many years in a long-term process only to declare that "*definitions of the health of an online community might vary depending on the type of community in question or type of project being jointly developed, how*

[42] Open Source Initiative

the injection of money into an online development community influences individual contributor behavior, and how individual decisions by contributors impact overall community health."[43]

In addition, infrastructural (programming) tools also influence the design of a community. Git[44], for example, is said to counteract the "internal ossification of power" (Benkler 2013). Whether textual data, packets, or bits, Git negotiates a multitude of "spatialities" that are built into its infrastructural working method, including hierarchical file systems, developer networks ("networks of trust"), the text space of the code, and the tree-like logic of consecutive commits, branches, and forks.

In this context, moments of articulation, transmission, mediation, and negotiation are the focus, with code, protocols, formats, and interfaces being positioned as topological operators[45].

The question of the social order in developer communities boils down to those decisions that a contributor can participate in and the effects of this individual coordination.
Forking as a practice has the capacity to serve as the "invisible hand of sustainability" that helps OSS projects survive extreme project events such as commercial takeovers and ensures that users and developers have the necessary tools to enable "change instead of decay".(Nyman et al. 2012)

> *"Informal structure also manifests itself in decisions that bypass hierarchy, divergences between formal & informal structure."*

Self-organization and governance models

> *"Governance norms in committers are commonly developed by participants that do not think them necessary, for a community that does not want them at the time. The result is frequently implicit, under-documented norms that increase barriers to entry for newcomers and allow incumbent contributors the instruments to derail unwanted decisions."*

It is the basic social order, not technical delegations, that can actually counteract an "internal ossification of power" (cf. Schrape): Formal governance[46] in projects includes, among other things, the modularization of the software, the assignment of roles to the contributors, the delegation of decision-making, training but sometimes also "indoctrination," as well as the provision of a uniform infrastructure to support the contributors.

There are usually sanction mechanisms in place to support the linking of incentives, cultural norms, and leadership roles that maintain coordination. The handling of complexity in this

[43] https://sloan.org/grant-detail/8434
[44] Git
[45] Topologische Operatoren
[46] FOSS Governance Collection

system depends on the properties of the software itself and the properties of the social and political structure in which it is embedded.

Stability is an important parameter in this regard. Metrics for analyzing aspects of OSS that can affect its long-term sustainability are composed of economic, social, and technical aspects. “Good” project management takes into account governance models, effective leadership structures, congruence with regard to the project goals, the key role often played by community managers, licensing models, and compliance.

In addition, for software systems with long life cycles, the ability of an OSS project to attract and retain new contributors is decisive for its long-term success.
A strong core of developers is necessary for the success of open-source projects, but not sufficient. Unless a broader group of “peripheral contributors” joins the core developers, the project cannot survive because it will lack the resources to find and fix bugs.

In case studies on a wide range of projects, a common set of challenges appears: difficulties in increasing participants and their contribution over the long term, difficulties in implementing organizational changes, and difficulties in making important decisions. There are also problems resolving intra-community conflicts and working out and enforcing the values of a non-homogeneous social group.

The literature on open-source communities, including Open Invention Network and Boehm (2019) repeatedly mentions these factors:

Welcoming, inviting culture
*“Meritocracy”**

- *Liberal contribution policies*
- *Equality of opportunities*
- *Useful, productive communities*
- *Ambitious hygiene factors*
- *Loyalty*
- And yet meritocratic governance cannot be equated with equality.

Today, most of the organizational structures of communities are only internalized by the initiators of the community and are not well documented for newcomers. The drifting apart of formal and informal organizations and the lack of supportive services in the form of community management are often not perceived as a relevant problem.
The resulting impact, exacerbated by the lack of positive competitive selection of leadership and the limited recruitment of contributors, is detrimental to the long-term growth and success of the developer community.

Contributions to “by-products” of the actual software are less valued than being directly committed at the product level. This can hinder effective specialization, as the various “roles” within the community are treated as having different merits. Similar to the organizational structure, there are also implicit status groups, roles, and privileges within the community.

Only a few projects reach the maturity phase and very few are integrated into operating system packages that are sold through traditional distribution channels. There is a difference between the value of popularity and vitality and even these are often undersupplied (Crowston et al. 2006).

Cave or cathedral: “Come for the technology, stay for the people”?

As already shown, there are many projects in the infrastructure sector in particular that only consist of a few people. These lack institutional isomorphism and the associated advantages (Werle 2003).

Significant contrasts can also be seen in the mix of motivational profiles between participants in community-based projects and those who work independently or on very small projects.

The subpopulations that contribute to large and very small projects varies in terms of their

- demographic characteristics
- educational qualifications
- prior experience
- participation in technically demanding projects
- the likelihood of receiving direct remuneration for work

The group of “experts” and “budding hackers” are more likely to be involved in larger projects, while the other subgroups (social learners, social programmers, user innovators) are associated with smaller projects (cf. also Sarma and Lam 2013).

The “romantic transfiguration” of open source mentioned above as a strategy for creating a community of qualified user innovators who voluntarily contribute to a process of collective creation as equals seems to be in a certain contrast to the indications of a clearly defined division of labor identified in the literature as standing between groups of core developers and a broader periphery.

In open-source projects there is often no formal organizational structure, at least initially in the project formation phase. Instead, personal motives, shared convictions, and values connect the virtual organization(s) with one another (Elliott and Scacchi 2003).

Governance processes are often established in the middle phases, in which the social process becomes more important than product aspects. Governance codifies norms in the informal social process that has already arisen (which is based on the composition of the community).

Volunteer contributors often show an explicit aversion to authority and formal decision-making. At the same time, they form the highest authority within their community. The feeling of personal creativity felt by contributors has the greatest influence on the amount of time they volunteer (Lakhani and Wolf 2003).

If one interprets the habits and practices of the volunteers in FOSS projects as a cultural phenomenon, the governance norms can be seen as the inside view of the respective community's culture.

How heterogeneity is dealt with, accounted for, or resisted in communities influences the stability, perseverance, and results of open-source development efforts.

Software as assemblage and the social formation of technologies: do artefacts have politics?

Technological processes and technologically mediated forms of organization have become commonplace. Not only since the emergence of workplace anthropology has it been assumed that language, (social) rituals, and even material environments such as workspaces (laboratories and hacker spaces, but also personal desk design and machine layouts) are inscribed in (user or developer-side) end products.

Implicit development environments as projects are linked to this motif and refer to the "invisible" structures that influence the development of software. Such prerequisites for production can be governance models, protocols, systems, or tools.

Another example is the choice of open source itself as an alternative political economy practice (with the limitations noted above). IDE necessarily also considers values, convictions, and potentials in the owners, developers, and supervisors of a particular project.

If we are able to dissect the non-material aspects of technology from its material basis, this puts us in the potential position of examining and operationalizing how technological change, resilience, and openness are reflected in infrastructural software projects against organizational changes, scaling, or adaptation and who or what is enabling these processes.

"Social shaping of technology" describes the factors, whether organizational, political, economic or cultural, that shape the design and implementation of technology.
This approach makes it possible to identify and analyze the socio-economic patterns that are embedded in the content of technologies (MacKenzie and Wajcman 1985; Bijker and Law 1992).

Each phase in the generation and implementation of technologies contains (see above) a number of choices among various technical options.
In addition to narrow “technical” considerations, a number of “social” factors influence the choice of options and thus the content of the technologies and their social effects.

This should not flatten the view of the dynamics around the development of software so much that it either suggests that these social factors alone determine the development of FOSS or that the market inevitably devours the software that should eat the world (Andreesen).
The challenge of viewing technology as a product of social shaping is problematized by Fidler and Badii (2019), among others, who address human rights considerations in working groups of the IETF and the difficulties in diagnosing the effect of such shaping *ex ante*.

In this respect, the important question remains to what extent the various traditional lines of open software development can trigger different path dependencies.

- Localization: SPOs as marketplaces
- Politics = twofold, of people and of artifacts
- History of the politics of artifacts: structural and functional similarities between politics and technology
- Technocratic thinking: social planning as the domain of politics, technology understood as the result of distributed market forces (Hayek), but today in the form of the idea: separate from political projects and intellectual lines from which they first emerged.
- “Code is law” (code is one of the few main sources of political order; others are law, norms, and the market)
- “Values in Design” school (including HRPC in the IETF) with the aim of researching whether standards and protocols in the broadest sense enable strengthen or threaten human rights = political properties.

  (Critique: assumes that context-independent values can be known in advance and thus coded, while reality instead gathers contextual, multi-causal thinking.)

Ideologies are definitely part of this immaterial part of software development and were recognized early on by influential figures in the scene.

Linus Torvalds once said during a discussion that he had used the proprietary tool BitKeeper within the Linux project.

> “*Quite frankly, I don’t _want_ people using Linux for ideological reasons.* I think ideology sucks. *This world would be a much better place if people had less ideology, and a whole*

lot more 'I do this because it's FUN and because others might find it useful, not because I got religion."[47]

Read the fucking manual: inclusion and exclusion in software projects

The "ideological foundation," especially in infrastructure-oriented projects, is the reason for the low penetration with enterprise developers.

New members in these projects are usually subjected to extensive examination procedures, in addition to the meritocratic economic logic of the "work brags for you." This principle often acts as a *proxy* (especially in the form of successful and publicly available commits) to save the time and energy that has to be expended in evaluating the substance of a person's abilities.

Since the participants (as a rule) take part in projects voluntarily, they generally feel entitled to determine their tasks themselves and to work free from of others' instructions (see Fogel / Benkler). Mechanisms aimed at enforcing conformity with social norms are mostly missing in the communities. Instead, most behavioral norms develop informally.

Even in classic peer production communities, however, hierarchical decision-making structures arise as their size increases. This results in power asymmetries and guidelines that ultimately contribute to the coherence of the projects and facilitate coordination.
As communities and project phases approach advanced levels of maturity, this cohesion is further established through manifestos or codes of conduct. Usually the cohesion decreases with the size of the group.

(Long-term) studies were carried out in various software projects that examined and modeled the development of communities and developer characteristics. They have covered such topics as the dimensions of user and dev-effort service quality, management structures, and loyalty to the OSS "ideology;" software characteristics (e.g. license conditions, target users, software modularity & quality); and community attributes (e.g. organizational support, financial support, trust & social network connections).

A Debian study finds that maintainers tend to remain committed to the project for long periods of time and that the percentage of volunteers in the project is likely to be higher than many software companies, which would have a clear impact on software maintenance (Michlmayr et al. 2007) .
Libre office, on the other hand, has a systematic approach to mentoring new contributors (organizational learning).

Guagnin (2017) provides an excellent portrayal of the tension between the fundamental opening of the design of software by its users and the productive closure of software development.

[47] Re: [PATCH] Remove Bitkeeper documentation from Linux tree

He also points out that epistemic regimes solve growth problems through resource allocation. “Read the fucking manual” is quite often the way developer communities deal with ignorance. In its early days, however, free software decidedly turned against the dichotomy of the difference in knowledge between developers of technology (the creators and guardians of rules) and its users who access these rules as a resource for action without necessarily having to understand them.

According to Coleman (2013), Debian was the first project that established transparent rules for an established meritocracy (community distribution). There is always a formal separation between developers and users: informal collaboration is desirable, but there is a quite a stringent process in place to gain official developer status.

Other infrastructure projects sometimes call themselves user-centric, rather than user-friendly. With reference to epistemic regimes in software, one could classify them as follows:

- Arch Linux deliberately sets itself apart from user-friendliness and offers a minimal base system with maximum configuration options.
- Ubuntu has a large number of lay developers, but Canonical makes the decisive decisions in the project.

Arch Linux stands for the “old-school” niche of free software development while Ubuntu represents the pragmatic use of this niche and its formation into a usable product.

Motivation and value models: ideologies, metrics, meritocracy

> *“Subcultures may emerge, ideology is a vehicle of clan control.”*

Lindenberg argues in Lakhani and Wolf (2003) that principled action is also a form of intrinsic motivation.

“Ideologies” can sometimes have a regulating role in software communities, as indicated in Bergquist and Ljungberg (2001), Markus (2007) and Raymond (1999).
This is because it can influence the effectiveness of open-source development teams, especially in the absence of formal controls:

> *“Adherence to some ideological components can be beneficial to the effectiveness of the team in terms of attracting and retaining input - but detrimental to the output of the team”* (Stewart and Gosain 2006).

OSS Norms	Illustrative Quotes from Open Source Narratives
1. Forking – there is a norm against forking a project, which refers to splitting the project into two or more projects developed separately.	…`forking' almost never happens. Splits in major projects have been rare, and always accompanied by re-labeling and a large volume of public self-justification. It is clear that, in such cases as the GNU Emacs/XEmacs split, or the gcc/egcs split, or the various fissionings of the BSD splinter groups, that the splitters felt they were going against a fairly powerful community norm (Raymond 1998)
2. Distribution - there is a norm against distributing code changes without going through the proper channels.	The taboos of a culture throw its norms into sharp relief. Distributing changes to a project without the cooperation of the moderators is frowned upon, except in special cases like essentially trivial porting fixes. (Raymond 1998)
3. Named Credit – There is a norm against removing someone's name from a project without that person's consent.	Removing a person's name from a project history, credits or maintainer list is absolutely *not done* without the person's explicit consent. (Raymond 1998)
OSS Beliefs	
4. Code Quality – open source development methods produce better code than closed source	Perhaps in the end the open-source culture will triumph not because cooperation is morally right or software 'hoarding' is morally wrong [...] but simply because the closed-source world cannot win an evolutionary arms race with open-source communities that can put orders of magnitude more skilled time into a problem. (Raymond 2000)
5. Software Freedom – outcomes are better when code is freely available	Restrictions on the distribution and modification of the program cannot facilitate its use. They can only interfere. So the effect can only be negative. (Stallman 1992)

The definition of ideology in our context is a common, relatively cohesive set of emotionally charged beliefs, values, and norms that hold a group of people together and help them understand their world.

Beliefs refer to the understanding of causal relationships, values are preferences for certain behaviors towards others, and norms are behavioral expectations. Although this stance is highly controversial these days, free software also came about historically as a “culture,” not just as a development strategy.

Kelty (2013) indicates that FOSS software is a boundary object that converts immaterial values and principles into material objects:
“It turns the negotiation of values into a subfield of maintenance, system administration, development & design.”

Values and norms are visible in at least four dimensions in the collaborative and open production of software:

- In the governance structures that projects create for themselves and the roles within the project structures
- Sometimes these policies are accepted in the technical artefacts themselves
- In the “marketing strategy” which frames FOSS as commercial products in terms of security and adaptability, but generates this values basis from the original idea of free software
- In terms of “added value” in their relationship with the markets

The socio-political discourse analysis[48] as a method assumes that ideologies are both developed and expressed in language and communication.

Narratives of contributing to a greater good are a well-known trope in FOSS development. Examples include:

- promoting technical innovation
- developing competitors to dominant proprietary goods
- advocating the social benefit of software freedom
- bringing together specialized technical needs and ethical convictions

During the specific literature research on value systems in software cultures, there are initial findings that could inform further research.

Dholakia et al. (2004) argue that norms for identification in virtual environments are important because they are most accessible or deducible on the basis of the archived communication of the group. Analysis and understanding of this communication facilitate the classification of preferences for potential new members of the group.

Scarbrough (1990) suggests that ideologies are analogous to maps and indicate the terrain (= belief systems), the goals (= values), and the signposts (= norms) that the actors use for orientation.

The identification of the most important ideological components in OSS teams and the different ways in which they influence the project results thus provides a provisional map for organizations that want to promote or understand the development of OSS.

And yet the values orientation is also subject to constant change: while respondents in studies often emphasize ideological motives for the initial development of FOSS, later the emphasis is more on practical reasons for choosing a particular project.
Less than half of the developers surveyed named the importance and visibility of a project as reasons for joining, while around ⅔ named the technical attractiveness and 80% the expected personal benefit of the selected projects.

Criticality and next fields of action, "market failure," and public interest technologies

Open-source software is a remix[49] in the sense used by Lawrence Lessig. A software project not only includes the code of its own repository, but also all of its *dependencies*, which are often tied to a specific version.

[48] Discourse analysis
[49] Remix (book)

With increasing product maturity, the dependencies in FOSS become more numerous and denser, the complexity of the system increases, as does the scope of risk management and compliance. As Benthall et al. (2016) describe, for the many projects lack licenses (23% of GitHub with 1,000 or more stars), there is no public version-controlled repository or similar vulnerabilities.

Security in software systems is achieved by managing the flow of information across various architectural levels of trust. The basis for this is the traceability of versioning, reporting systems, best practices in code quality, and security analysis measures.

Open-source software carries an additional risk: community failure. And only a mix of methods can be proposed as an antidote to this security gap.

There are entities and interventions that already address certain project needs (e.g. legal protection, organizational development, public relations). Apart from Software in the Public Interest or KDE e.V. (or CCT), for example, large software foundations often address conflicts or challenges in such a way that they are resolved in line with the interests of the market.

This literature review has already shown that market logic does not necessarily ensure sustainability in projects.

Summary and Outlook

The market deploys its own laws and logics in which "interest" appears, for example, in the term "interest rate". This interest is due to repay capital advances that someone has taken out in order to be able to afford what is available as a “limited good”.

From an economic point of view, software is an information good which, in its proprietary form, shows characteristics of a club good, but as FOSS, those of a public good - non-exclusive, non-rivalrous.
Moreover, information asymmetry is usually particularly high in this good. Information asymmetry exists when information about a good is unequally distributed between the supply and demand sides.

Even if the source code is disclosed, a great deal of effort and special expertise are required to derive product properties from it. Software products therefore fall into the category of experience goods and trust goods.
For suppliers of goods with distinctive experience and trust properties, there are generally extensive opportunities for strategic behavior (Linde 2008).
This is especially true for the club good "proprietary software"; however, with the incorporation of free software elements into all layers of the layer model, the "providers" of open code are also affected by commoditization, as has already been shown in previous chapters.

Effectively, for many companies operating with a FOSS strategy, it is a matter of outsourcing parts of the value chain when they rely on FOSS or provide FOSS themselves: Formerly private costs for development and maintenance (of the company) are externalized, socialized, whereas the skimming of profit from the entire value chain remains predominantly with the company.

The economic peculiarities of network goods, which take full effect in software, have led to a high degree of concentration in an unregulated market. Market shares of over 90 percent in certain segments are not unusual (see above), and market shares of more than two-thirds are common.
Markets with such high concentration tend to bring about undesirable competitive outcomes. As a consequence, resources are misdirected, costs caused by the supplier are not borne by the supplier and parts of the demand are not catered to appropriately.

FOSS as a combination of technology, law and social processes for the production of software in the information society, based on division of labor, with a fully developed free rider[50] problem and as a boundary object[51] is an area in economics in which this market failure is currently visible.

Market failure is a typical call of action to regulation for the state, but also for the tertiary sector.
Third sector institutions between market and state can be understood as subsidiary non-market social institutions. They arise from the fact that not all needs are adequately met by the market, or that certain goods and services are provided by non-profit rather than for-profit organizations - these actors include software foundations, family foundations and organized civil society with civic tech volunteers as potential grantees.

Public goods per se fall under the government mandate to regulate "market failure"[52] to prevent consumer detriment: While there is no question that tangible infrastructure is part of services of general interest, this is not yet sufficiently defined for intangible infrastructure as seen in software.
“Daseinsvorsorge” or: services of general interest refer to the state's sovereign task of providing goods and services that are necessary for human existence. These include, among other things, energy and water supply, transport services, telecommunications, broadcasting, street cleaning, and sewage and waste disposal, at moderate prices and of comparable quality.
Digitization not only permeates people's everyday lives, practices and commodities, but has also long been a basis of public services, as described.
The software relevant in this regard is therefore immaterial infrastructure nowadays.

The concept of public interest allows for an approach to safeguard of (FOSS) software quality and production cycles, among other things, through tax contributions:

---

[50] Free-rider problem
[51] Boundary object & Sebastian Gießmann, Klassifizieren und Improvisieren. Ein Kommentar zu Geoffrey C. Bowker und Susan Leigh Star
[52] Marktversagen | bpb

Public interest as the core of democratic political theories stands for the "level playing field", a safeguarding of fundamental rights of citizens and their participation.
The "right to the Internet" is often mentioned in related discussions.
However, this term would have to be extended to include the reliability, security and availability of the software that is based on the haptic network structure - and enables society to function.

The first initiatives in this direction often come from cities and municipalities, which, especially in Germany, are also the executors of material services of general interest as the executors of local self-governance. Following the "Public Money, Public Code"[53] campaign, for example, a common public repository for FOSS administrative software is being discussed currently.
Efforts[54] are underway to increase funding guidelines and knowledge of software infrastructure among funders and to join forces in (digital) civil society.

In the future, this should be promoted more systematically in order to better address known vulnerabilities of FOSS and to diversify investments[55].
Similar to merit goods, the public good thus becomes a politically desired public good whose provision depends on a political consensus and whose determination is the subject of political debate.

No longer does the diagnosis of a market failure exclusively determine what may constitute a public good. The philosophy of the state determines whether citizens' socio-political goals are to be classified as worthy of promotion in the sense of a public good.
A public good or a merit good expresses a society's self-understanding. It is a sign of political will, be it in the form of intra-societal or international solidarity, be it in the form of social valuations in favor of certain goods or ideas of equality, be it in the form of common socio-political goals or of a public or collective decision-making process.

The question of what markets are responsible for is also determined by society through controlling institutions, state regulations and frameworks.

This literature review has attempted to describe the interplay between individual motivations, organizational practices, and institutional structures in mobilizing and coordinating resources and governing community-based production.

It was able to show that open source is dependent on the internalized values of individuals, the economic logic of a distinctive production process, and a set of social and political structures that maintain coordination and manage complexity.

Many of these processes are implicit and cannot be directly controlled or influenced from the outside.

---

[53] Public Money, Public Code
[54] Gute Software: Wie schaffen wir die Zukunft, die wir brauchen? - D3
[55] Edit Policy: Wo bleibt Europas Open Technology Fund?

However, there is a potential of legal and organizational innovations and stabilizers that need to be explored in more detail in the future in order to then implement targeted measures that preserve the "exceptional good" of open source software, especially in its positive aspects for society.

Open software is playing an increasingly important role, also politically. This importance should be taken into account both economically and in terms of public recognition.