Editorial: Announcing The Journal of Open Law, Technology & Society

DOI: 10.5033/jolts.v11i1.134

by The Editorial Committee, coordinated by Shane Coughlan.[a]

(a) OpenChain Project Director, Linux Foundation.

Abstract
This editorial article by the Editorial Committee of the Journal of Open Law, Technology & Society announces the change in name, and expansion in scope, of the International Free and Open Source Software Law Review.

Keywords
Law; information technology; governance; Free and Open Source Software; International Free and Open Source Software Law Review; Journal of Open Law, Technology & Society; JOLTS.

When the International Free and Open Source Software Law Review was launched in 2009, we stated that it was our "hope and expectation that it will provide a centre of excellence for the very best in analysis of issues facing users and advisors in the development, deployment and governance of Free and Open Source software...."[1] Since then, the International Free and Open Source Software Law Review has had a storied history in the realm of free and open licensing around software technology. While the law review was initially inherently biased towards a relatively small audience, we have been fortunate to both gain readership and provide value to this audience over a period of more than 10 years. Many things have changed since we published our first issue in 2009. At that time, open source was still gaining traction in the business and legal spheres. Notwithstanding the exceptional success of service companies such as IBM and HP in the preceding decade, and the continued rise of "open source" companies such as Red Hat, in 2009 general awareness across multiple market segments of free and open source software and the licensing models behind it were still relatively low. That is no longer the case. The rise and tremendous success of companies like Google, Facebook and Uber is predicated on a foundation of free and open software. Platforms once unobtainable to all but the largest companies are now within the reach of even the smallest start-up, and we are seeing a terrific explosion of new services, products and business models as a consequence.

Free and open source software has moved from "emerging" to "market dominant." Indeed, free and open source software has been such a success that it has inspired an adjacent explosion of broader and increasingly visible open hardware, open access, open data and open knowledge initiatives across

1 Mitchell, Iain G. (2009) 'Foreword and statement of purpose: an introduction to IFOSS L. Rev.', IFOSS L. Rev, 1(1), p. 5.

the globe. In acknowledgement and in celebration of this expansion of open models, the editorial committee of IFOSSLR is taking steps to position our publication and our community for continued relevance to technical, business and legal professionals operating in these expanded areas. The freedom and openness that brings success in markets, in revenue and in social development is no longer confined to software and neither should this journal be so confined. We are delighted to announce an expansion of our scope and a further broadening of our audience and community. Welcome to JOLTS, the Journal of Open Law, Technology & Society, a publication intended to help you stay abreast of the most interesting and most relevant topics emerging in our sphere. We look forward to continued collaboration with you and with your peers.

About the author

Shane Coughlan *is the OpenChain Project Director for the Linux Foundation and an editor of this Journal.*

The emergence of governance norms in volunteer-driven open source communities

Mirko Boehm[a]

(a) Visiting Lecturer, Chair of Innovation Economics, Technical University of Berlin; Director, Linux System Definition, Open Invention Network

DOI: 10.5033/jolts.v11i1.131

Abstract
Free and open source software communities develop their governance norms and practises as they grow from small to medium to large sized social groups. Communities with a small number of participants typically organise informally. As the community grows, the need for coordination grows as well and at some point, a more structured organisation becomes necessary.
The growth stages are defined by the coordination mechanisms applied – ad-hoc coordination for the initial small group, consensus focused auto-organisation for the medium sized group, and structured, more formalised coordination for the large sized group.
The main interest of the communities is to attract and retain contributors and to facilitate contributions to their products. The communities studied in this qualitative embedded multiple-case study, exhibit governance related debates and conflicts, as they reached a large size, leading to difficulties in further growing the number of involved contributors and sustaining the community activities.
The paper researches the emergence of governance norms in these communities and the role these norms, once established, play in the management of the communities in their then current stage.
The study finds that the governance norms in communities 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.
The paper focuses on the essential contradiction between the communities' aim to maintain devolved authority at the contributor level and a requirement for effective decision making and policing mechanisms to implement and maintain that.
It recommends that communities, instead of deferring or down-playing the need to set up explicit governance norms, purposefully develop norms that explicitly define structure and processes so that they support, enforce and protect the devolved authority their participants should have and encourages new participants.

Keywords
Free and Open Source Software; governance; code of conduct; meritocracy; FSFE; KDE Community; Wikimedia

1. An inside view on social norms in communities

On February 3, 2016, something happened in the KDE free and open source software (FOSS) community that would become the dominant topic of discussion for more than 6 months: An announcement[1] was sent to the community mailing list that a draft of a new vision for the community was being worked on. This announcement triggered close to 350 postings to various mailing list threads, constituting almost half of all discussions within the community in the first half of 2016. It led to heated discussions between drafters of competing visions, public endorsements and statements of support, virtual ad-hominem attacks and even contributors leaving the community in anger. How could an announcement of something so basic, so fundamental to a large decentralised group of volunteers, like a vision, create such distress?

In May 2016, a code of conduct for the Free Software Foundation Europe (FSFE) was announced to the organisation's coordinators. After small changes, a version was approved and sent to the core team for a decision to be made in June. After just a few people voiced an opposition to some of the wording, the process came to a halt. When it was picked up again in October of the same year and circulated in an almost unchanged form to the same people who had seen it already in May-June, it spawned one of the fiercest debates in the history of FSFE with more than 200 mails in just two weeks. Suddenly, people spoke up in opposition not only to the content of the code of conduct but about the very need for a code of conduct in general. Indeed, the code of conduct had, in the eyes of some of the participants, become a tool not to include contributors but to silence unwanted opinions. While the general consensus seems to have been in favour of adopting a code of conduct, the process came to a halt again, since no decision could be reached. The code of conduct was finally adopted without substantial changes in October 2017.

There are many instances of such soul-searching in FOSS communities as they reach maturity and achieve a large number of contributors. It can be observed that these controversies focus on questions regarding how the communities internally manage their social norms and questions of community governance, which form the totality of implicit and explicit behavioural norms, codes and processes that regulate the relationship between contributors and the community. While these are certainly not the only challenges the communities face as they grow, evolving community governance appears to be a particularly difficult problem for each community to manage.

When interpreting the habits and practises of voluntary contributor collaboration in FOSS communities as a cultural phenomenon, governance norms are seen as the inside view on the culture of that particular community. They are an outward expression of the way the communities see themselves. Understanding this inside view as to how the communities are expected to operate is relevant not only regarding issues of community management, but also to outsiders as the basis of the views held by that community's contributors and how to engage with that community. To establish successful collaboration with these communities, the public, regulators, businesses and influencers of technical innovation such as standards development organizations (SDOs) or the patent offices would be well-advised to understand the cultural norms and practises of these FOSS communities.

This paper researches the governance norms that have evolved in volunteer-driven FOSS communities as they grow from an initiative of a few contributors to large and often international organisations. Assuming that these norms are based on the aggregate of the individual convictions

1 https://mail.kde.org/pipermail/kde-community/2016q1/002241.html

and expectations of those contributing to the community, the paper describes this inside view that the communities have of themselves and in particular the behaviour the actors engaged in the community expect from their fellow contributors, from the community as a whole and from outsiders – individuals, other organisations and the public.

2. Governance in communities with voluntary participation

For this study, the work of the wider Open Source community is primarily viewed as a social process producing information as a public good. It is a knowledge-intensive process that has inputs in the form of labour (the work of the contributors) and capital (the funding required by the communities). The output is information goods, most prominently the software components that are freely distributed to the public. The nature of free software licenses makes them "non-excludable" and "non-rivalrous" and therefore "public goods".

The production of a public information good is one key element that defines a FOSS community. The other key element is voluntary participation of the contributors in the community. The understanding of FOSS community applied in this paper is that of a social group of contributors that participate voluntarily in the production of public information goods.[2] The participants in these groups, the contributors (see section 2.2), collectively create the community's products and make them available to the general public by distributing them under a free or open source software license. This study focuses on communities that consist predominantly of individual volunteer contributors.

Communities that grow beyond a very small group of contributors develop (sometimes unconsciously) functional specialisation between their contributors, division of labour between formally and informally defined subgroups, and integration of the individual contributions into an overall product. They become organisations. Functions that contributors specialise in can be product related (software development or content creation in general, "maintainership" over submodules, or release management) or support (marketing and public relations, finance, event management and legal). To successfully release products over time, communities need to coordinate the work of the individual contributors so that, through a repetitive process of content creation, gatekeeping and filtering for quality, integration and distribution, the product improves over time.[3] Coordination in this context is understood as a process, not as a task performed by a manager. With respect to the production process, the *need for community governance* results from the necessity to coordinate the work of a diverse group of volunteers to create the community product.

From an outside perspective, of users or the general public, the communities are mainly known for the products they create. Potential contributors want to engage with the community based on the product related participation opportunities, and on what is generally known about the culture of the community. A common recommendation is to "treat every user as a potential volunteer".[4] Most contributors participate in a community for a limited period of time, leading to fluctuation in participation. To grow the number of incoming contributions, communities need to attract new contributors and retain the existing ones, so that the difference between influx and outflow remains positive. With regard to the interaction with the outside world, the *need for community governance* results from the necessity to maintain and grow the contributor base that forms the community.

Even though FOSS communities commonly operate as decentralised self-organised groups, they develop elaborate informal and formal rules and practises for their social process. These rules and

2 Albert O. Hirschman. The Passions and the Interests. 20 Anv Sub. Princeton University Press, Jan. 1997.
3 Yochai Benkler. "Coase's Penguin, or, Linux and "The Nature of the Firm"". In: The Yale Law Journal 112.3 (Dec. 2002), pp. 369+.
4 Karl Fogel. Producing Open Source Software: How to Run a Successful Free Software Project. O'Reilly Media, Inc., Oct. 2005.

practises are referred to as the *governance norms* of the community. A social norm is "a prescribed guide for conduct or action which is generally complied with by the members of a society".[5] The term governance "refers to all processes of social organisation and social coordination"[6] within groups. It describes the processes of governing a formal or informal organisation performed by a formal government, a market or a network. Governance is expressed through a wide range of instruments ranging from laws to social norms, as well as language and culture. Any social group that coordinates working together towards a common goal will exhibit some form of governance. Whereas *government* refers to the institutions that exert power and influence over a constituency, governance can exist without institutions. Communities often hesitate to develop formal governing structure. This directs the focus of analysis on governance (a process) over government (the structure) when studying FOSS communities.

The subject of governance can be considered to be decision making and conflict resolution within the social group, by using broad definitions for both terms. "Decisions" is used here in the sense that whenever a small subset of the contributors or the whole community jointly agree on a course of action on some subject, a decision is made. Similarly, "conflict" is understood broadly as any disagreement of one or a group of contributors with any action or decision made by another subgroup or the community as a whole. Decisions do not need to be made in a formalised process, nor does conflict require a formal complaint or a heated argument. Decision making and conflict resolution are essential elements of collective action.[7] How the organisation defines who may participate in what decisions as a community member, and what organs form the organisational structure, characterises key aspects of that governance.

It can be assumed that organisations exist to further the common interests of their members.[8] The reason for FOSS communities to exist is to facilitate the interests and motivations of their contributors. To illustrate the governance of FOSS organisations, this study will review the reasons why the organisations exist, the organisational structure of the community, the processes by which decisions are made and challenged, and how and with which roles contributors participate in them.

The ethics and convictions of the individual contributors should be reflected in the organisation's vision and mission statements. The formal and informal organisational structure provides the framework for the community's production process. Constitutional documents like bylaws and manifestos establish formal structure. Representative bodies like boards, committees and working groups are the most visible formalised form of it. Formal structure projects authority by assigning decision making power to individuals or organisational units. In addition to those, informal structures that are more difficult to identify are likely to exist. Informal structure manifests itself in decisions that bypass hierarchy, or in strong impact of the opinions of individuals that are not appointed to representative positions. FOSS communities commonly show a preference for minimal formal organisation (see section 2.4), which leads to the assumption that informal structure has a more preferable effect than usual. Formal organisation is also more difficult to change, since it typically requires both a qualified majority of the group members and a conscious effort to understand and reconsider the current structure and identify how it should be changed.

One potential reason for a perceived need for organisational change is a divergence between the formal and informal structure. Opposition to reform indicates that group members may be more comfortable with the existing balance of formal and informal structure. Decision making processes and conflict resolution mechanisms define how decisions are initiated and then made, and how to appeal against or escalate them in cases of disagreement, how decisions will be implemented or enforced, and how the community deals with minority opinions and opposition, especially in the case

5 Edna Ullmann-Margalit. The emergence of norms. Clarendon Press, 1977.
6 Mark Bevir. Governance - a very short introduction. Oxford University Press, 2012.
7 Russell Hardin. Collective Action. RFF Press, June 1982.
8 Olson Mancur. The Logic of Collective Action: Public Goods and the Theory of Groups. Revised. Harvard economic studies, v. 124. Harvard University Press, Jan. 1965.

of controversial decisions.

Decision making processes correspond to organisational structure in that commonly, paths of escalation or appeal follow the hierarchy of formal organisation. Conflict resolution is directly related to decision making processes as the cause of a conflict is either the wish for a decision to be made, or to appeal against one that was made. The balance between decision making processes, instruments of appeal and conflict resolution is what enables contributors to influence the community production process.

The social order within the community defines which stakeholders can take part in what group decisions. Differentiation can for example be based on eligibility or group status. A regular contributor may not be eligible to take part in a board decision or may not have the status to take over a maintainer role. The question of social order in communities boils down to what decisions a contributor can participate in, and what the impact of the individual vote is. It relates to the definition of group membership that separates insiders from outsiders, but also possibly to the status of groups within the community. It is also related to how contributions are valued and translate into merit and recognition for the contributor. There may be a sense of equality, or a sense of elitism where "only the core contributors should have a say". If social order in communities is considered important, there should be well-defined processes on how to gain access to those status groups within the community that carry weight in important decisions.

FOSS communities sometimes discount the importance of decision-making or claim that decisions are made or conflicts resolved by "the wider community", and that therefore organisational theory does not apply. This argument however does not hold water, since it cannot reasonably be disputed that communities delimit members from outsiders, have status groups, make decisions and resolve conflicts (even if those elements are not all made explicit).[9] By analysing organisational structure, decision making processes and the community social order as key elements of governance, it is possible to compare communities even if they create unrelated products.

2.1. Growth stages of communities

The differentiation between the inside and the outside view of the community's social process puts the emphasis on the demarcation of the social group, in as much as it defines who is a member of the group and who is an outsider. Being an insider means accepting the group's rules, providing influence and in turn expecting to participate in the group's governance. Being an outsider leaves a choice to either interact with the community and accepting its norms, or to abstain from interacting with it. The community is afforded the same choice not to engage with an outsider based on how compatible their actions are with the group's norms.

Since about 2010, participation in FOSS activities as a phenomenon has changed from an exotic movement to a common mode of operation in the ICT industry.[10] This suggests that communities have also matured into established organisations with solidified cultural norms and values. The communities studied in this report have all existed for longer than a decade. They will be viewed as mature and stable organisations where processes can be observed through the activities within their formal and informal structure. Their norms and values have developed over time as a result of the interaction between community participants who join the group voluntarily out of their own motivations, and the community as an organisation of its own, which creates structure and processes according to the goals of the group and the strategies chosen to reach them.

9 Amitai Etzioni. "Two Approaches to Organizational Analysis: A Critique and a Suggestion". In: Administrative Science Quarterly 5.2 (1960), pp. 257–278.

10 Jeff Licquia and Amanda McPherson. A $5 Billion Value: Estimating the Total Development Cost of Linux Foundation's Collaborative Projects. Tech. rep. https://www.linux.com/publications/estimating-total-development-cost-linux-foundations-collaborative-projects. The Linux Foundation.

The communities will be investigated at three different growth stages: The point of foundation called the *initial stage*, the time when the group has reached a small to medium number of contributors (typically between 20 and 50 active contributors) called the *medium stage*, and a *late stage* with a large number of community members (often more than 100). The growth stages are defined by the coordination mechanisms applied to the social process, which show different characteristics in these different stages of development.

At the time that a particular FOSS initiative is formed and in its initial stage the goals and motivations of the group of founders and of the initiative as a whole are identical. There is commonly great enthusiasm about the joint initiative. The original authors publish their work and communicate that contributions from others are welcome and appreciated. More contributors join and participate out of a motivation similar to the motivations of the original authors – to contribute to the product, make it available to the public under a FOSS license, and rely on the community to keep the process going. As long as the group is small enough for *ad-hoc coordination*, the subsequent contributors joining will find themselves in a similar situation. It can be assumed that the participants in the initial stage will be homogenous in their motivations, cultural backgrounds and interests. Worries about governance usually do not exist.[11] [12] [13]

Interests and motivations will start to diverge as the community grows and matures. The group will reach the medium stage when the number of participants becomes too large for ad-hoc coordination and changes into a form of consensus focused auto-organisation. At this stage, deviations between individual expectations and community behaviour exist. Instead of relying on formal structure in the organisation, the communities rely on a consensus-driven, participative debate culture. Disagreements will be discussed at length until a resolution is achieved. The resolution does not necessarily require consensus or a formal decision.

The KDE community, for example, applies a method called “lazy consensus”, in which contributors have begun to work on their favoured solution while alternative courses of action are still being discussed. The direction the community later prefers can then be decided based on the results of the discussion and on the experience from the work already provided by its contributors. Other communities apply similar mechanisms that prefer product related contributions over “bureaucracy”. It is apparent that such mechanisms rely on close cohesion of the group’s participants, a low grade of specialisation amongst the contributors and a relatively small number of stakeholders in the decisions. Not only are the communities themselves content with such informal self-coordination, they also develop a strong preference for the absence of formal structure. Since contributors participate voluntarily, they feel entitled to self-identification of tasks and to work free from direction given by others.[14] While it may cause friction, self-identification contributes to the allocation efficiency of peer-production processes.[15]

The transition into the late stage of community development is commonly marked by more formalisation. Communities may establish internal working groups to facilitate contributions to more specialised topics. To coordinate with external partners, they may nominate community members to represent the community in their committees. To account for these delegated responsibilities, the representatives may be required to report on their work on a frequent basis at regular meetings. In general, more functional differentiation occurs between the community participants. Delegation of power and responsibility becomes more pronounced, leading to a more prominent role for the

11 Eric S. Raymond. The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary (O’Reilly Linux). O’Reilly, Oct. 1999.
12 Karim Lakhani and Robert G. Wolf. “Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects”. In: Social Science Research Network Working Paper Series (Sept. 2003).
13 Steven Weber. The Success of Open Source. Harvard University Press, Apr. 2004.
14 Karl Fogel. Producing Open Source Software: How to Run a Successful Free Software Project. O’Reilly Media, Inc., Oct. 2005.
15 Yochai Benkler. “Coase’s Penguin, or, Linux and ”The Nature of the Firm””. In: The Yale Law Journal 112.3 (Dec. 2002), pp. 369+.

community leaders.

At this point leadership positions that previously more or less fell to those who volunteered to speak to the press or be elected to the board become more prestigious. Appointments carry more weight and elections for them grow competitive. The differentiation of roles within the community enables jockeying for position and a sense of entitlement, especially as regards long-standing contributors. Once this formal organisational structure is established, the community shows behavioural patterns similar to other larger common good oriented community organisations like unions, sports clubs or cultural initiatives. Being a part of the community becomes a motivation in its own right, complementing the motivation to contribute to the community product directly. Matters of procedure and community management attract more attention. A share of the collective energy of the community is redirected inwards to discuss the community itself. At the same time, behavioural norms are still in place that developed during the early and medium stage. For example, communities have established a “break all the rules” rule that postulates that every participant is free to decide the best course of action, even if it means ignoring a norm or rule. Or there may be a “who does the work decides” rule which postulates that those who take part in the debate should not interfere with those working directly on the community product.

Based on these considerations, it can be expected that communities in the initial stage require almost no coordination, communities in the medium stage rely on organic self-coordination, and communities in the late stage act more in accordance with the logic of collective action in large groups.[16] The transition into the late stage should necessitate a change of the effective community governance norms away from informal mechanisms of the medium stage towards more explicit, formal mechanisms appropriate for larger-scale collective provision processes. The intense governance-related conflicts and debates that accompany the shift of the communities into the late stage indicate that this change did not fully happen in the cases studied in this report.

Intense inner social conflicts indicate a divergence between the individual expectations of contributors and the group norms developed by the community. These conflicts may be resolved positively, resulting in a re-alignment of individual and group motivation. However, if the conflict is too severe or for other reasons cannot be resolved satisfactorily, it may also lead to either individual contributors deciding not to participate in the group anymore, or the conflict may cause a fork, where the group splits into two that continue to develop towards the initial goal separately.[17] Forks are rare, as substantial effort must be invested to create a competing community organisation. More commonly, contributors defect if their perception of the quality of the community diminishes. Since there is no centralised resource planning, defections may go unnoticed. It is difficult to assess the impact of individual decisions or the design of decision-making processes on the contributor base. Sometimes communities prefer not to make any decisions to avoid losing contributors, which results in indecision manifested for example in bike-shedding debates.[18]

2.2. Community composition

Entities participating in FOSS initiatives can be either individual volunteers, organisations (participating directly or through contributions of their employees) or staff employed by the community. This mix is referred to as *community composition*. Most communities consist of individual volunteers and employees of businesses, with a very small share of employed staff.[19]

16 Olson Mancur. The Logic of Collective Action: Public Goods and the Theory of Groups. Revised. Harvard economic studies, v. 124. Harvard University Press, Jan. 1965.

17 Gregorio Robles and Jesu´s M. Gonz´alez-Barahona. “A Comprehensive Study of Software Forks: Dates, Reasons and Outcomes”. In: Open Source Systems: Long-Term Sustainability. Ed. by Imed Hammouda et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–14.

18 Karl Fogel. Producing Open Source Software: How to Run a Successful Free Software Project. O’Reilly Media, Inc., Oct. 2005.

19 Dirk Riehle et al. “Paid vs. Volunteer Work in Open Source”. In: System Sciences (HICSS), 2014 47th Hawaii International Conference on. IEEE, Jan. 2014, pp. 3286–3295.

This study focuses on communities that are made up predominantly by individual volunteer contributors. These communities, like KDE, may distinguish between contributions of time and effort spent by an individual contributor and the financial contributions made by businesses. The individual contributors are able to become personal members in KDE e.V. even if they contribute during work time. The businesses employing them may only gain "supporting membership" through which they support funding the organisation by paying a membership fee, but do not attain a vote on a board or in the annual general assembly.

FSFE similarly does not allow other organisations to take part in their activities directly. These rules underline the significance "volunteer driven" communities associate with individual contributions and their aversion towards any form of institutional investment. This norm of only valuing individual contributions builds upon the expectation that the community production process should be steered with regard to product quality alone, and not influenced by interests of external parties. In the case of KDE e.V., this norm is explicitly codified in the bylaws of the organisation, which only accepts individuals as members, not legal entities.

In communities with a majority of contributors who are employed by businesses, like the Linux kernel developer community, the reputation of companies is more closely related to the aggregated contributions of their employees.[20] Businesses and individuals participate in FOSS activities for different reasons. Individual volunteers are mainly intrinsically motivated through a sense of achievement and personal enjoyment. Signalling of key skills to potential employers also plays a role.[21] Businesses on the other hand, are motivated by economic rewards and the opportunity to influence. For example, participation gives them the opportunity to create non-differentiating components in their products in collaboration with other parties with similar interests at drastically reduced research and development costs, as well as participation transaction cost.[22] Businesses also benefit from their FOSS activities being a source of quality staff and promoting a healthy innovation ecosystem.

Depending on community composition, the communities develop norms and principles that reflect the specific mix of motivations of their constituency. This opens up a continuum with purely volunteer driven communities on one end, purely business driven communities on the other end, and mixed or hybrid communities in between.

The majority of FOSS communities are hybrids, resulting in a set of norms and practises within those communities that reflect the motivation of both organisational and individual contributors.[23] We expect that the norms and principles adopted by the communities can be clustered based on the contributor composition, and that communities with relatively similar contributor structures develop relatively similar norms and practises. To facilitate separate analysis of these sets of motivations, this paper focuses on studying communities that are (almost) exclusively made up of individual volunteer contributors. These communities would be expected to have developed comparable governance norms.

20 Jonathan Corbet and Greg Kroah-Hartman. Linux Kernel Development, 25th Anniversary Edition. Tech. rep. http://go.linuxfoundation.org/linux-kernel-development-report-2016 . Linux Foundation.
21 Karim Lakhani and Robert G. Wolf. "Why Hackers Do What They Do: Un- derstanding Motivation and Effort in Free/Open Source Software Projects". In: Social Science Research Network Working Paper Series (Sept. 2003).
22 Johan Lin˚aker et al. "How Firms Adapt and Interact in Open Source Ecosys- tems: Analyzing Stakeholder Influence and Collaboration Patterns". In: Re- quirements Engineering: Foundation for Software Quality: 22nd International Working Conference, REFSQ 2016, Gothenburg, Sweden, March 14-17, 2016, Proceedings. Ed. by Maya Daneva and Oscar Pastor. Cham: Springer Inter- national Publishing, 2016, pp. 63–81.
23 Sonali K. Shah. "Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development". In: Management Science 52.7 (2006), pp. 1000–1014.

2.3. Separation of open source products and community processes

FOSS communities create freely available products in a social process of peer production.[24] While it is a common expectation that producing a product under a free software license goes hand-in-hand with applying a transparent, open process, based on voluntary participation, this is not always the case.[25] There are FOSS products that are produced by a single vendor in a closed process and without relevant participation of other parties.[26] Some products are developed by a single, dominant commercial vendor where outside participants are required to grant rights to relicense the product proprietarily to the commercial vendor through some form of contribution agreement. These agreements do not reduce the freedoms provided by the product license, but they change the community process from decentralised to centralised.[27]

Other products, like the Linux kernel, are built by a decentralised community and do not require any attribution of rights. The licensing of FOSS products on the one hand and the community processes applied to produce them on the other need to be considered separately. While the choice of license defines whether or not a product is free or open source software, the governance norms applied by a community determine openness.[28]

We assume that the preference in a community for a more or less open governance model correlates closely with community composition (see section 2.2), and that volunteer driven development communities have a strong preference towards openness and transparency in their governance.

The two main schools of thought about the essence of FOSS represent these two aspects separately as well:

Some proponents of the term "open source" put more significance on whether or not a product is distributed under a FOSS license approved by the Open Source Initiative. They see software released under a free license as a means to an end.

Others who put more emphasis on software freedom consider the work of communities to be part of a political movement representing a cultural shift that works towards a world without proprietary software, with an ethical underpinning. The FSF for example argues that "software should not have owners".[29] The separate product and process aspects of FOSS however are present and relevant in both schools of thought.

2.4. Voluntary participation and meritocracy

Both camps agree that contributors form the community by taking part in the production process voluntarily and without direct compensation for their efforts. Communities with a small number of contributors are typically organised in an informal way and work coherently. As the number of contributors grows, the difficulties of informal organisation grow until they reach a level that requires a more formal structure. There is however no authority in a position to impose such a structure.

The raison d'ˆetre of enterprises and institutions is commonly defined ex-ante by, for example,

24 Yochai Benkler. "Coase's Penguin, or, Linux and "The Nature of the Firm"". In: The Yale Law Journal 112.3 (Dec. 2002), pp. 369+.
25 Josh Lerner and Jean Triole. "The Simple Economics of Open Source". In: National Bureau of Economic Research Working Paper Series (Mar. 2000), pp. 7600+.
26 Dirk Riehle. "The single-vendor commercial open course business model". In: Information Systems and E-Business Management (Nov. 2010), pp. 1–13.
27 Karl Fogel. Producing Open Source Software: How to Run a Successful Free Software Project. O'Reilly Media, Inc., Oct. 2005.
28 Liz Laffan. "A New Way of Measuring Openness: The Open Governance Index". In: Technology Innovation Management Review 2 (2012), pp. 18–24.
29 Richard M. Stallman and Lawrence Lessig. Free software, free society: selected essays of Richard M. Stallman. SoHo Books, 2010.

investors, regulation or government, which act as a form of higher power that imposes a purpose on the entity.

Like sovereign states, the question of the purpose of a FOSS community is self-referential. A community exists to serve the interests of the participants, which participants are also the community. Where states resort to postulating a constitution which then anchors acts of government, communities develop governance mechanisms based on voluntary participation and meritocracy (unlike in the original satirical understanding of meritocracy, the wider Open Source community historically uses the term with an overall positive connotation, which has recently been challenged).[30]

Voluntary participation in the group means that an acceptance and implementation of group decisions needs to be achieved based on a common and mutual understanding. Since they are contributing voluntarily, participants expect to have peer status in the group and influence according to the value of their contributions. This is what FOSS communities refer to when they call themselves meritocratic. When formal structure emerges, the principles and norms applied typically reflect voluntary participation and meritocratic peer status as well. From this, two important collective action issues are derived that communities struggle with, that of decision making, and that of enforcing conformity to social norms.

Decision making is difficult as the winning majority has no instrument to force those who disagree with the decision to implement it. There is no individual cost involved in simply ignoring a community decision. Communities therefore prefer to reach consensus and do so in a discussion process that may be laborious to reach a decision, even if only with a very qualified majority.

They intentionally refrain from allowing the same question to be re-raised after a decision is made, without reason or a great deal of effort from the participant re-raising it. Some communities explicitly acknowledge the difficulty of making formal decisions and relegate them to the status of opinion polls (Wikimedia) or restricting the use of votes to the acceptance of new members (KDE).

The sensitivity of making decisions that are not based on consensus reflects the importance of attracting and retaining contributors and underlines the social process aspect of community activity. This sometimes results in a separation of administrative leadership and product related decision making. For example, KDE e.V. manages KDE’s assets and funds, but by way of a community principle may not interfere with product related technical decisions.

Mechanisms that aim to enforce conformity to social norms are mostly absent in communities. Initially most behavioural norms develop informally. In the medium and late stages, community manifestos or a code of conduct may be put in place. At this stage, the necessity for a formal rule that restricts how community members may behave may be questioned. For example in KDE and FSFE.

The communities studied did not build effective means of actively influencing behaviour towards the expected outcomes. While in early stages this need is mitigated by the strong cohesion of the group, in later stages the lack of it is often seen as an obstacle to developing more diversity.[31] Based on anecdotal evidence from the interviews, the necessity for explicit behavioural rules is typically questioned by long-standing community members that are part of the dominant social group within the community. The aversion to enforcement of rules is related to the self-referential nature of communities. Critics of explicit rules often question where the authority would come from to enforce them.

30 Michael Dunlop Young. The rise of the meritocracy, 1870-2033: The new elite of our social revolution. Vol. 85. Random House, 1959.
31 http://rachelnabors.com/2015/09/01/code-of-conduct/, but also https://modelviewculture.com/pieces/a-code-of-conduct-is-not-enough

Governance based on voluntary participation and meritocracy is an essential attribute of volunteer-driven communities and stems from the self-referential nature of the community's purpose. While communities that are led by a "self-appointed benevolent dictator for live" exist, the cohesion of these communities depends on this individual maintaining a strong meritocratic status.[32] Separating product outputs from community process can also provide guidance as to which FOSS producing organisations should be considered communities: a community distributes products that through applying a FOSS license are public goods *and* the associated organisation will create this product in a process that is based on voluntary participation.

When applying these criteria to classifying organisations into FOSS communities and others, the cases studied in this report meet this requirement. A single entity that produces FOSS without voluntary participation of others, like Google developing Android, is a valuable FOSS contributor, but not a community.

2.5. Study design and method

The aim of this study is to describe and understand in detail how and what influences the emergence of governance norms in volunteer-driven FOSS communities, and what effect these norms have on the community as it grows from a new initiative to a large size organisation. A qualitative, embedded, multiple-case study of the inside view of social norms in three communities was performed. It may be considered a mixed method study design that combines the multiple-case study with theoretical modelling based on personal observation and experience, however that personal experience is also embedded within the same cases.[33]

The cases analysed in this study are large, mature, successful volunteer driven FOSS communities. There are only a small number of communities that achieved this level of success over an extended period of time – probably about a dozen. Another key criterion for selecting the case studies was access to individual key community actors and the organisations' decision-making bodies. The author has access to internal information of some communities because of his own history as a long-term contributor. As there is only a small number of communities that reach the late growth stage, and these communities develop a strong cohesion and a distinct insider culture, an analysis from outside these organisations could not lead achieve the same level of understanding of how the communities function. The qualitative study design explains the interpretive, experience based, situational character of the cases, and facilitates analysis of organisational development as a long-running, episodic and evolving phenomenon. A small number of community cases were chosen to avoid stereotypical generalisation caused by an unwarranted higher level of aggregation.[34] The decision in favour of a qualitative research approach was supported by the assumption that quantitative methods do not promise reliable insights given such small constituencies. Experiments also were not considered feasible.

The study was conducted by performing 16 interviews with long-standing contributors to the communities who were either founders or who rose to community leadership positions at a later stage. Some of them are still active in these communities today. Some have resigned from their functions. Overall, the interviewees who contributed to this study represent more than 200 person-years of FOSS community leadership experience. The interviews gathered information about the personal ethics and convictions of the contributors and their interpretation of how the community governance norms and organisational design have developed.

The interview concept was developed against the theoretical framework (see section 2) which was built upon the individual experience of the author and the current state of FOSS community

32 Eric S. Raymond. The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary (O'Reilly Linux). O'Reilly, Oct. 1999.
33 Robert K. Yin. Case Study Research: Design and Methods. 3rd. New York: SAGE Publications, Inc, Dec. 2003.
34 Robert E. Stake. Qualitative Research: Studying How Things Work. 1st ed. New York: The Guilford Press, Mar. 2010.

governance research. The interviews provided evidence that puts the experiential expectations into perspective.

The interviews helped obtain unique interpretations held by the interviewees, to aggregate information from many interviews, and to access the personal experience of the interviewees that would otherwise be unobservable. The interviewees provided data about their own individual positions and their interpretation of questions about the community as a whole. By connecting observational insights with understanding of the individual ethics and convictions and observations on community governance by the interviewees, expectations in the theoretical framework about community governance norms and contributors expectations can be tested.[35] The final report is based on personal observation, the interview results and information available in artefacts like minutes from general meetings, statutes, manifests or codes of conduct that the communities published.

The qualitative method chosen leads to limitations in the applicability of the study's results. The emphasis on the inside view regarding governance norms does not take external factors like the explosive growth of FOSS into account or considered the fact that this may also have contributed to community growth and other development trends.

Demographical changes affect communities – one interviewee mentioned a perceived decline in the proclivity of individuals to volunteer for social causes. Market trends that affect the position of the community's products also probably play a role. More importantly, the subjective, personal, constructivist point of view applied in the study means that observations only represent the experience or interpretation of the participants and the author, not necessarily a true meaning. The findings in this report can therefore not be generalised. They should however provide a valuable deep understanding of the inside view the communities and contributors have on themselves.

3. The mindset behind community governance

The interviews for this study consisted of three separate parts. The first one focused on the individual contributors, what their expectations and convictions were and why they joined their communities, how these expectations developed or changed over time, and what principles or ethics of individual conduct are important to them. The second part of the interview focused on the community as a whole, and the third part discussed inner-community conflicts as focal points for governance debates.

This section is based on the first part of the interviews. Some of the key governance documents like organisational statutes, the codes of conduct or community manifests have been authored by the interviewees. It is assumed that since the interviewees are founders or long-time participants in the communities and through their leadership roles actively influenced the community constitutions, their expectations and convictions strongly influenced the emerging governance norms. Even if these may have changed at a later point in time, their influence should still be apparent.

3.1. Engage in a community of makers

It is commonly assumed that participation in the development of FOSS products is primarily need-driven.[36] However, the need for a solution to a particular problem does not explain sustained investment of effort into being a community member in good standing. To justify this behaviour, being part of a community requires that additional rewards like a sense of belonging are generated. The most limiting factor to contributors is the time available to undertake such intrinsically

35 Ibid.
36 Eric S. Raymond. The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary (O'Reilly Linux). O'Reilly, Oct. 1999.

motivated work. The different projects compete for this time from individuals.[37] Being an integral member of a community means sharing time available between creating contributions to the product, which is perceived as a fun, productive and creative activity, and progressing through a social hierarchy involving engaging in community processes, which may be considered a necessary but time-consuming overhead. The initial question to ask is why contributors that end up being involved over an extended period of time and investing significant amounts of their personal time into FOSS contributions make the decision to join the community in the first place.

All participants in the study stated that the impetus to engage with the community and become a part of it was to contribute to the community's main product. They also stated that the creation of that product needed to be a positive challenge to motivative them and not be a routine task. Only a third of interviewees initially chose a specific community because of its social norms. A strong majority however said that over time, being a part of the social group became more and more important to them. A common phrase used to describe this phenomenon is "come for the technology, stay for the people."

The freedom to choose what task to work on in a group of like-minded people and the creativity in the search for a solution that this affords was mentioned by almost all interviewees as a motivator for becoming part of the community. This indicates that contributors are attracted to a FOSS community because of the challenging products it sets out to create, and only then learn about the social norms within the community and begin feeling attached to them. Those interviewees who are not active in their communities anymore usually exited gradually, reducing the amount of their contributions over time until they stopped. The expectations on an individual's productivity are in line with existing research which identified the sense of personal creativity felt by the contributor as the biggest impact on contributed hours.[38]

Multiple interviewees mentioned that they felt the community mission was "worth fighting for" in that it combines a productive, creative activity with a sense of contributing to a greater good, like fostering technical innovation, building up competition to dominant proprietary products or advocating for the societal benefits of software freedom. The communities provide a virtual place where individuals who share this combination of rather specialised concrete need and ethical conviction congregate. Whilst this may readily exist online, it may not occur in a physical location which is of course less likely to reach a critical mass and become a gathering place for like-minded people.

FOSS communities are meritocratic in the sense that individuals gain influence solely based on their contributions to the combination of community product and social process. This environment attracts highly skilled individuals who interviewees felt they could look up to and learn from, but at the same time who accepted them as equals. Such learning is a rare opportunity not commonly available to highly skilled individuals in physical environments. Meritocratic peer status based on concrete contributions also leads to a notable absence of other forms of discrimination by for example nationality, race, gender, age or other factors, at least initially. Individuals with non-binary sexual orientation are a common sight at community events, and do not usually attract much attention. One interviewee assumed a higher-than-normal share of individuals with symptoms of autism or Asperger syndrome amongst the contributors.

3.2. Equality of opportunity among peers

The interviewees joined their communities when they were still in the initial or medium stages. Some explained the perceived group size as "tiny" or mentioned that there was a positively motivating

37 Karim Lakhani and Robert G. Wolf. "Why Hackers Do What They Do: Un- derstanding Motivation and Effort in Free/Open Source Software Projects". In: Social Science Research Network Working Paper Series (Sept. 2003).
38 Ibid.

"David-vs-Goliath" feeling to working towards the group's goals. Most explicit and implicit governance norms were established in these phases. If communities needed to provide a combination of productive contribution opportunities and matching ethical convictions, what were the expectations of the contributors when they joined in respect of how these communities should operate?

Only a minority of the interviewees joined their communities with expectations of their governance norms. Some of the community processes in fact came as a surprise to newcomers, for example the extent to which new participants were immediately accepted into the group and even encouraged to represent the project at community and other events. Some intentionally joined the community to learn about how it works and stayed in an observer role for a period of time. Most participants developed their preferences towards governance norms while being a contributor.

Most of the interviewees emphasised a priority of "doing" over "talking". Contributors to the KDE community are very conscious of the "who does the work decides" rule. While an absence of discrimination is generally expected, the meritocratic rule within the communities does not translate to egalitarianism. Participants earn their prestige or even the right to participate in debates within the community through the contributions they make. This translates to an expectation of *equality of opportunity*, but not of an equality of rights. Ideally, the status any contributor might obtain depends on how much effort she or he invests into contributing to the community's causes. Community members who "only talk" found little acceptance and were sometimes explicitly denied a voice in debates. Some stated that initially they felt like the community needed only the grass-roots meritocracy structure, but that in later stages they changed their mind about that.

Almost all interviewees mentioned an inherent tendency to form sub-groups within communities specialising in particular functions or product aspects. These sub-groups remained at a size that continued to allow for ad-hoc coordination, even as the overall community grew beyond a size where this would be effective. The governance within these sub-groups was less standardised, one interviewee described them as "little villages with chieftains". Sub-groups also helped to retain regional or cultural cohesion and the sense of productivity by isolating their members from what some described as excessive debate. Because they initially associated themselves with one of the sub-groups, the communities felt smaller to the interviewees at the time they joined than they really were in numbers of overall participants.

Surprisingly, the fact that the community product is distributed under a free software license or generally is a common good was not mentioned as an expectation by the interviewees, but as a precondition. Similarly, the absence of discrimination is expected as a given. Some said they would simply not consider participating in any initiative unless the outcome is freely available to all.

3.3. Balance of makers and community builders

The communities in this study all succeeded in establishing themselves as important in their respective fields and grew from the initial stage to the medium stage within two to four years. Almost all interviewees mentioned that being a member of the community became a goal in itself. Where previously, community membership was a means to facilitate contributions to a product, the contributors built personal attachments to the community as a sort of virtual home, where they maintained friendships and developed loyalty to the group. Some of the early contributors quickly rose to community leadership positions that became an important part of their self-perceived identity. They reallocated a share or all of their available time to community management tasks, reducing their product contributions in the process. Differentiation emerges between the product developers as the makers and the community builders as the maintainers.

Most participants could not rely on previous experience in managing larger communities and were surprised by their own success. One interviewee involved in Wikimedia explained how the explosive growth of the community in 2004 caused the group to consider how to coordinate "once we were more than three". The communities struggled with the transition from the initial stage. For multiple interviewees this transition happened when they realised that they did not know all contributors personally anymore, which also indicates a breakdown of ad-hoc coordination. The KDE core team retreated into private invite-only mailing lists where "those who do the work" could coordinate. The need for organisation and administration manifested itself when the community started to organise the first all-hands conferences.

To manage funds and donations the legal entity of KDE e.V. was created. The reliance of private communication channels was felt to contradict the expectation of transparency and open process by one interviewee. However, it was considered necessary at the time, and is still in place today.

Contributors that mainly work on community building face a dilemma that only becomes apparent over time: while they are contributing to core functions of community management by being board members or project representatives, they are not taking part in product development anymore in a community that primarily exists for that purpose. Consequently, their merit eventually declines. Some cling to their influential positions, possibly realising that they will not be able to maintain their community status once they hand over to a successor. Progressively they disconnect from the product-focused elements of the community and begin to value procedural questions about community management higher than facilitating the productive processes. Instead of being supportive, the administrative entities created by the communities exhibit a tendency to grow to be *antagonists* to the community of makers, "with members who contribute little and a board that contributes even less", as one interviewee described it.

3.4. An ambitious, productive meritocracy of equals

Over the years that the participants in the study contributed to FOSS, it can be assumed that it is common for them, once in a while, consciously or sub-consciously, to take a step back and to reconsider whether their time and money spent for the community's purposes is still a worthwhile investment. We asked them what criteria they apply when evaluating the perceived quality of their community. The answers were surprisingly uniform across the participants from all three communities.

All or almost all interviewees agreed to the following criteria:

The communities need to provide a *welcoming, inviting culture*. It forms the basis for the close social connection that develops between contributors. The communities should also extend trust to newcomers, allowing them to learn the community norms even if it involves making mistakes. This includes flat hierarchies for contributing to the community products. The communities implement an "open door policy", as KDE puts it, where newcomers, once they have an account, have access to almost all of the project's infrastructure. Common well-accepted exceptions are system administration, legal and financial functions.

Participants expect their communities to implement *meritocracy*. While the understanding of meritocracy is not completely uniform, regarding what constitutes a contribution and how it should be valued, the prestige and influence of contributors within the community should be measured by the aggregated value of their contributions, and nothing else. Two aspects of meritocracy are less defined in this regard, how merit diminishes over time (forcing old-timers to make way for new contributors), and which other soft factors like socialisation, being in the right place at the right time, gender or age, influence merit. More recently, *liberal contribution policies* as applied by the Node.js project address these issues.[39] Experienced contributors consider meritocracy in FOSS communities

39 Mikael Rogers. Growing a contributor base in modern open source. 2016. url: https://opensource.com/life/16/5/growing-

very important but question the naïve understanding of meritocracy that is commonly applied.

In this respect, there is an expectation of *equality of opportunity*. All communities, even if they down-play it, form status groups like administrators, formal members of the organisations or elected positions, which have noticeable barriers to entry. A common expectation is that all these positions should be open to anybody willing to contribute enough to the community cause, subject to a common set of criteria. Equality of opportunity is different from meritocracy in that status and merit may diverge. Individuals who attained status based on past merit that has now diminished may still be influential in the organisation. Similarly, a valuable contributor may have merit but not advance in status because, for example, no elected positions are available at that time, or old-timers get elected to them. This may cause personal disappointment, possibly disengage valuable contributors and eventually cause them to spread negativity or even leave the community.

Contributors are looking to contribute to *useful, productive communities*. They want their contributions to help the community to get closer to achieving its goals. It is often not enough to contribute to the product, contributors also expect the product overall to be useful, and to receive feedback or even to get more contributions from users outside the community. One of the main values that the community adds to the peer production process is to add distribution and communication channels to attract users to the community products and create a feedback cycle back to product development.[40] An increase in the required share of available time being spent on community-internal debate detracts from the sense of productivity.

On top of the community helping them to be useful and productive, contributors expect the community's mission and vision to be *ambitious*. It is not enough in the long term to "build a better mouse trap", as the intention to build a FOSS replacement for a proprietary product has been described. Achieving societal change towards software freedom by lobbying for it is considered an ambitious goal, as is freeing a large user bases from lock-in to proprietary products, or creating "a world in which every single human being can freely share in the sum of all knowledge", as in the Wikimedia vision. In an abstract sense, contributors want the community to aim at making the world a better place, and for their contributions to help with that.

Next to these four quality criteria, the interviewees mentioned aspects that pose preconditions for engaging with a community. These preconditions may be considered *hygiene factors*, criteria that do not positively motivate contributors, but whose absence would be considered a reason not to engage with the community at all.[41] Such factors include that community products are public goods, an absence of discrimination, a positive communication culture or code of conduct, respect for minorities, reasonable escalation mechanisms, supportive technical infrastructure, and opportunities for learning and personal improvement. These factors are "basics that need to be there".

We asked the participants if they felt a sense of responsibility for or *loyalty* to the community as they progressed, which they unanimously agreed they did. Some felt that the team they worked with started depending on them, and even tried to empower their colleagues to reduce that dependency. It would have felt bad for them to leave the community while this dependency existed. The merit they attained and the personal relationships built with other community members gives them a sense of responsibility for the community as a whole. They also understood that it would be hard for them to replicate the time and effort invested, which imposes a cost on exit that makes it difficult for long-term contributors to leave the community.[42]

contributor-base-modern-open-source (visited on 03/13/2019).

40 Yochai Benkler. "Coase's Penguin, or, Linux and "The Nature of the Firm"". In: The Yale Law Journal 112.3 (Dec. 2002), pp. 369+.

41 F. Herzberg, B. Mausner, and B. B. Snyderman. The Motivation to Work. Transaction Publishers, Jan. 1993.

42 A. O. Hirschman. Exit, Voice, and Loyalty: Responses to Decline in Firms, Organizations, and States. New edition. Harvard University Press, July 1970.

The communities struggle with continuing to be inviting to newcomers. Both product- and process-related barriers to newcomers emerge and grow over time. In the initial phase, all contributors are newcomers and mistakes are commonly seen as part of the process. Later, longer-term contributors have accumulated experience, and the quality gap between contributions by them and by newcomers widens. The German language Wikipedia for example introduced a pre-publication review ("*Sichtungsprinzip*"), which reduces the trust extended to new contributors. This reduces the feeling of appreciation and acceptance that the participants reported they themselves felt when they started contributing. Interviewees said that the trust that contributions are generally valuable has been lost to some older community members.

3.5. Ethical principles applicable to community governance

Contributors will only be intrinsically motivated to voluntarily spend significant efforts in a social group that conducts its activities in a way that agrees with the ethical convictions and principles of the individual. Such convictions are formed through life and rarely change. They can be considered an external variable that the governance norms of the community should reflect. We asked the interviewees which principles that are considered "just" in other social groups, they think, should also be applied in their communities.

The interviewees strongly agreed that working code, meritocracy, solidarity and transparency are key principles that they look for in the governance of their community. *Working code* refers to the expectation that "code should speak louder than words", meaning that concrete code or other contributions to the community product should be valued higher than "politics". This argument is related to a general paradigm that postulates that FOSS development should focus on delivering a working implementation over, for example, writing a detailed specification.[43] They feel that the work on the product should be the benchmark by which the community is judged. This principle is important to contributors because it describes very directly how the communities should operate.

Meritocracy is mentioned again as an individual expectation, indicating that the term is not only used to describe a mechanism of community management, for example in codes of conduct, but also as an expectation of a norm that directly influences the motivation of individuals to contribute. Communities implement meritocracy because their contributors expect them to or would otherwise not participate.

Solidarity is a principle that shows itself in an extension of trust to newcomers and more experienced contributors, a belief in their generally good intentions, and a habit of mutual support. It is part of the fabric of the social cohesion that the communities form and enables them to overcome otherwise separating attributes like race, gender, nationality or age. Tensions in debates have often been resolved in good humour by invoking Hanlon's razor, reminding everybody involved not to attribute to malice what can be adequately explained by (collective) stupidity.

Transparency is a common expectation that should result in processes and debates that are accessible equally to and documented for all contributors. This is understood as an invitation to participate, not a duty. The transparency principle is to a large extent engrained in the technical infrastructure of projects. Discussions take place on mailing lists, wikis or online chats, and are commonly logged or otherwise preserved. Activities are coordinated in project management tools or task trackers, often in ways similar to how a software development project would be organised. This habit may be encouraged by familiarity with software engineering tools. Interviewees from all three communities mentioned that they feel like their organisation is not as transparent as it should be with regard to governance processes, as opposed to product contributions.

There is no agreement on whether or not communities eventually need to fall back to *majority*

43 Steven Weber. The Success of Open Source. Harvard University Press, Apr. 2004.

decisions. Some interviewees believe that if the community cannot reach consensus on a subject, it is better if no decision is made at all. Others accept that situations exist where making a decision is inevitable. When asked directly, most but not all would prefer a decision over non-decision. All understand that with voluntary participation decisions cannot force contributors to act in a certain way. However, there was also disagreement over whether or not consensus should be sought as the decision-making principle. There is little awareness that, for many issues, staying with the status quo is one of the alternatives to choose from, and that *not* making a decision is equivalent to deciding to stick with the status quo. The consequences of decisions are evaluated, but those of indecision are commonly not.

The expectations regarding decision making processes appear to change over time, with multiple interviewees reporting that during their early involvement their preferences have leaned much more towards unstructured ad-hoc coordination, while after being involved for a number of years they feel that better defined and documented decision-making processes and especially escalation and conflict resolution mechanisms become necessary. It is not clear if this change is caused by gaining more personal experience, or by the communities outgrowing the initial and medium stages and operating as larger organisations.

The concept of transparency is connected to a number of common expectations regarding what constitutes a FOSS development process. Habits like open technical discussions, online collaboration and releasing working code early and regularly are considered strengths of the wider Open Source community. To enable that, a contributor or newcomer should be able to understand what the community is working on and how to take part in it based on information available online. Contributors also need to ensure that they possess all rights to use, study, modify and distribute the community product without a need for later negotiation. The emphasis on transparency is born out of the necessity to facilitate distributed collaboration in a diverse team.

There is an understanding that the communities implement these principles well with in the production processes, but not so well in community governance. In particular, a lack of transparency and meritocracy is noticeable in the decision making of the community leadership and in staffing high ranking community functions. In terms of documented structures and processes, the communities do not differentiate between product and governance related decisions, despite the fact that many of the norms applied rely on the fact that technical changes can easily be reverted.

The interviewees mentioned that there is a close match between their personal ethics and convictions and the social norms they expect the communities to develop. The fact that the governance of the communities is modelled so closely after their ideal of how an organisation that benefits the common good should operate is a strong motivator for them to continue contributing.

4. Case studies

The communities studied for this report are primarily volunteer driven (their contributors are *amateurs* in that their community engagement does not constitute a significant direct source of income, as opposed to professionals), mature (they have been working towards their purpose for multiple years), comparatively large (they have attracted between dozens and hundreds of contributors over time) and successful (each of them is recognised as an influential organisation in their respective field). Even though all of them produce freely-licensed public goods, the communities differ in the nature of their main product: the KDE community primarily produces software with a focus on end-user needs, FSFE is a free software pressure group that advocates the benefits of software freedom and Wikipedia produces an online encyclopaedia. The community product is the key element that provides participants with the opportunity to contribute. However, the communities have been selected using the hypothesis that community composition has a more dominant impact on governance norms as they emerge than the nature of the main community product being created.

Governance norms are expected to develop according to the expectations and convictions of the contributors in respect of the characteristics of the collaborative peer production process.

The following section analyses the vision and mission, the formal and informal organisational structure, the decision making and conflict resolution mechanisms and the rules for group membership of each of the studied communities in each community.

4.1. FSFE

4.1.1. Mission, foundation and history

FSFE was founded in 2001 with the mission of bringing about sustainable change towards societal freedom in the use of digital technologies. The ambitious "life-time scale"[44] of this mission was understood as comparable to initiating "a second enlightenment" with regard to software freedom in Europe. At the time of foundation, FSFE was considered the European sister organisation to the FSF. The organisation gained recognition by representing the wider Open Source community in the anti-trust case against Microsoft's dominant position as a supplier of operating systems for personal computers in the European Commission. FSFE also represented the community at the World Intellectual Property Organisation and the Internet Governance Forum.

FSFE introduced a fiduciary licensing program in 2003 that allows FOSS contributors to have their copyright ownership managed by the organisation. The FLA program strengthened the role of FSFE as a representative organisation of the European free software community.

In 2005, FSFE launched its "fellowship" program, widening the base of supporters to those who wished to contribute to the organisation's purpose financially, instead of or as well as by investing personal time. The fellowship had a limited representation in the general assembly through two seats for finance contributing fellowship representatives, until 2018, when the fellowship program was terminated.

With the increasing adoption of FOSS in commercial products, the complexity of compliance with the free software copyright licenses and the danger of free-riding behaviour of some manufacturers became apparent. With support from external parties and in cooperation with gpl-violations.org, FSFE launched the Freedom Task Force, an initiative intended to help contributors and businesses to create and use software distributed under FOSS licenses correctly. The European Legal Network was founded in 2008 as a venue for legal and technical experts to collaborate on legal and licensing issues related to free software and quickly expanded beyond Europe through the support of key lawyers in the European community. The Legal Network is currently the single largest network of free software legal experts world-wide.

FSFE continues to grow in influence and size. Today, it wields relevant political influence at the European and EU member state level, has strong backing from the FOSS community, and hosts the most influential legal and licensing discussions globally. It employs a president as well as a small group of policy analysts, campaigners and administrative staff.

FSFE offers opportunities for FOSS activists to participate in a small set of well-defined key products - political influence on the regulatory framework relevant for free software, coordination of various regional free software related activities, and facilitating the discussion and promotion of free software

44 Quotes in this section are taken from the interviews, unless otherwise noted.

legal and licensing topics.

4.1.2. Formal and informal organisational structure and conventions

The original intention of FSFE's organisational structure was a federated system of regional chapters with a central coordinating office that represented the organisation at the European and global level. Based on subsidiarity, the local chapters would be autonomous except when central coordination is needed. The concept of local chapters did however not materialise, with only one ever being active. The idea of local chapters was eventually dropped in 2016, leaving the organisation with a headquarters in Berlin that represents FSFE across Europe.

The legal entity of FSFE is an "eingetragener Verein", a charitable association registered in Germany. The statutes are the most explicit documentation of FSFE's structure and processes. The formal members of the organisation are somewhat misleadingly called the general assembly, even though it is a permanent decision-making organ. The general assembly elects the president and vice president. Historically, the president has been the most visible and influential role in the organisation. The president, vice president and treasurer together form the executive council, the de-facto day-to-day decision-making body of the organisation. Activities are coordinated between the FSFE team, the group that comprises all active contributors with separate communication channels and a formalised, email-based decision-making process. The general assembly and the staff are subsets of the team. Occasionally, task groups are set up to handle specific topics.

Informally, especially in the initial stage of FSFE, some individuals exercised significant influence without a formal mandate and were called "luminaries" by some interviewees. Since FSFE was founded as a sister organisation with the "blessing" of FSF, early activities where coordinated with FSF, and approval was sought for key political positions and messaging. The influence of FSF waned over time, also because FSFE applied a more collaborative style of governance than FSF. Approval of general assembly membership is handled very selectively. There are no term limits or requirements of re-election for general assembly members. Early-stage participants still wield significant influence in the general assembly, even though some would not be considered part of the team today since they are not actively contributing. The approval for full membership in the organisation is selective and depends on a combination of individual initiative and pull from existing members. As of May 2017, there have been 27 full members, with about two thirds actively contributing in the past 6 months.

To some interviewees, the formal organisational structure no longer reflects reality. They consider the loosely defined *team* as the core of the organisation, since most of the day-to-day work is coordinated amongst them today. The team however does not have authority over budgetary or executive decisions that are a prerogative of the general assembly, marking a significant deviation between power and responsibility. Many conventions are implicitly defined and passed on by word of mouth. Long-standing rules may still be in effect but are not very well known or followed. Some norms and processes are clearly under-documented, which one of the FSFE founders during the interviews classified as a "rookie mistake".

Due to its history and initial community composition, FSFE has a regional concentration in Switzerland and Germany, with the head office being located in Berlin. Since almost all activities are conducted online, the impact of the community is spread relatively equally across Europe. Local (country) teams exist in 8 European countries as of January 2019 and, in the case of the European Legal Network, globally.[45]

45 https://fsfe.org/about/localteams.en.html

4.1.3. Decision making and conflict resolution

Most decisions are made at the FSFE team level in a consensus-driven, mainly email-based process. An issue is raised on the team mailing list. After deliberation a proposal is submitted to a specific decision mailing list. The proposal is accepted if no rejections are raised. In the case of objections, the proposal is returned to discussion, refined and re-submitted in an updated form. While this process could theoretically repeat multiple times, the consensus-driven culture within the group limits iterations so that almost no proposal reaches a third round of debate. If the team realises that consensus cannot be achieved, the proposal falls back to the president who then may abandon the proposal or, if considered necessary, force a decision. There is awareness of the need to find resolutions that are palatable to those who raised objections. The value of the decision is weighted against the cost of demotivating members of the team.

The general assembly decides based on a simple majority after detailed deliberation of an issue. Only a few more strategic decisions are made at the general assembly level. The local groups develop their own processes that are not prescribed by the central office. This approach works well since those groups are small in size.

In general, there is no defined way to appeal against a decision. Staff are supposed to direct complaints against decisions to the executive council, in which the president is one of three members. There is no process to appeal against general assembly decisions.

Overall, these convoluted and circular rules of appeal potentially result in an absence of accountability across the organisation. This is balanced by the dominant motivation to work towards a common goal, however there are no protections against abuse. Compliance with norms and processes is effectively left to chance.

The decision-making process within the team and guidelines as to how the general assembly and the organisation as a whole should work were documented early on in FSFE. One of the interviewees assumed however that only a small fraction of those currently active in the organisation are fully aware of them. While staff and general assembly members may assume they are commonly understood, these documented norms and processes are not transparent to anyone outside the general assembly and organisation’s staff, and so create a barrier to newcomers’ effective participation . As a result, reforming the formal structure has proven to be very difficult.

4.1.4. Community membership, roles and privileges

Throughout its initial stages when FSFE represented the free software community in anti- trust cases a significant risk of elitism was felt by participants. Formal membership in the organisation was dependant on approval of the existing members and applied selectively. This risk continues to be perceived as relevant by some in relation to structural reform today. There are significant barriers to entry and a selective approval process to formal membership in FSFE.

The vast majority of FSFE contributors are not formal members of the organisation. Governance is indirectly affected by this as the strong influence of long-term contributors or staff as an unrepresentative membership can be used to influence which issues are put up for a community decision-making. Multiple attempts at organisational reform in recent years ended in indecision.

There is an ongoing argument as to the extent that the formal structure should influence the work of the community. While contributions to FSFE’s mission do not require formal status in the organisation, the lack of clarity regarding ways to participate and to gain access to key roles may have a detrimental effect on contributor engagement. Interviewees pointed out that contributors commonly slowly fade away instead of leaving with a clear end to their engagement. This makes it difficult to measure levels of engagement, like the number of active contributors or contributions made in a time frame. The effect of the social structure on the success of the communities is not explicit.

4.1.5. Structural reforms and outlook

Similar to the other case studies, the formal organisation of FSFE has rarely changed. The fellowship program was introduced in 2005. The fellowship seats in the general assembly combined with the fellowship representative elections existed from 2009 to 2018, as did the role of executive director. There have been no other major governance changes to the organisation since 2009.

A *strategy process* was started in 2013 and is still ongoing. It was designed as a top-down process and mainly involves the inner circle of general assembly members and staff. The wider Open Source community, particularly that outside of the regional focus area of German-speaking countries, is not involved. While the process produced statements of intent, it did not influence the day-to-day work of the community much or trigger an alignment of activities across the subgroups. This situation indicates a lack of forum for strategic discussions where all stakeholders engage, such as an actual "annual general assembly" across a wider membership. The issue is exacerbated by some members' position that FSFE is not accountable to the wider Open Source community and only speaks for its members. An interviewee summarised the strategy process as "a lot of discussion and very little results".

According to the interviewees, there is no systematic process to maintain and document the formal structure or to align it to the development of the informal one. Some rules are being ignored since the problem they anticipated, like a hostile takeover, has not occurred. To newcomers the governance of FSFE is hard to understand and not transparent. Contributors that are not part of the staff or the general assembly have practically no chance of influencing the organisation.

An interviewee mentioned a perceived lack of impetus for change since about 2011, with FSFE's leadership mainly taking on a maintainer role (see section 3.3). This is exacerbated by a difficulty in creating effective collaboration between staff and volunteers. There is a chance that the contributions of hired staff displace volunteer work by reducing intrinsic motivation. This may create a potential zero-sum scenario where spending on personnel changes whether FSFE receives contributions from staff versus volunteers but does not necessarily change the overall level of contributions.

In summary, the formal and informal organisation of FSFE as well as its decision-making processes appear to have been well thought out originally but have not been updated in pace with the growth of the community and outside changes. The well-thought out organisation design has aged and is now outdated and in need of reform. It appears that the main problem is not with the quality of the initial setup, but with the absence of constant, gradual improvements to it over time.

The KDE community took a different approach, but due to a comparable lack of a systematic improvement process, ended up in a similar situation.

4.2. The KDE community

The KDE project was founded in 1996 by Matthias Ettrich when he wrote to the de.comp.os.linux.misc Usenet group looking for contributors to a new, visually pleasing, easy to use graphical user environment for the increasingly popular Unix operating systems, named KDE. Unix was regarded as the superior operating system to Windows, but the usability of modern graphical desktop environments on Unix systems was a real, widely felt limitation at the time. The call for contributors fell on open ears in software development circles. There was a concrete desire in the early contributors to build the kind of desktop they wanted to use themselves. The motivation of the founders and that of the initial stage community was identical. The possiblility of competing with the dominant proprietary desktops produced by large enterprises was an important motivation. Within less than a year, a small group of early contributors produced a first version of the new desktop that showcased the enormous potential for innovation in this niche.

KDE released multiple successful versions of its main product, the “K Desktop Environment” until the present day– version 1.0 in 1998, 2.0 in 2000, 3.0 in 2003 and 4.0 2008.[46] By 2009, KDE had reached one million source code contributions, making it one of the largest FOSS projects at the time.[47]

After the release of version 4.0, KDE changed the mission of its community to be an umbrella organisation that supports various free software initiatives, the desktop, now named “Plasma”, being only one of them. In this move, KDE explicitly shifted towards emphasising the community process benefits over the development of the main product. Today, KDE is a large, still mainly volunteer-driven community with multiple regional subgroups (for example in Latin America and India). It still develops the desktop and other products and is networked and affiliated with the Open Source Initiative, FSFE and other FOSS stakeholders.

4.2.1. Formal and informal organisational structure and conventions

In November 1997 KDE e.V. was registered with the mission of representing the budding KDE community in legal and financial matters. It gained charitable status in 2012. The bylaws of KDE e.V. were the first and for a long time the only written constitutional document of the KDE community. They were also for the large part copy-pasted from unknown sources and not tailored to the needs of community collaboration. When the growing numbers of contributors led to difficulties in coordination, the core contributors, instead of building an overall structure supportive for the coordination of a larger group, retired into more specialised communication channels. Almost all other behavioural norms at the time have been implicitly assumed. In August 2003, an updated version of the bylaws drafted specifically to support the work of the community was approved. It introduced the concept of passive membership that enabled long-term contributors to remain members of the organisation when their level of involvement declined, without endangering voting quora, and codified the invite-only acceptance criteria for individual membership.[48]

Membership in KDE e.V. is reserved for active contributors to the project, who need to be invited to membership in the organisation by two existing members, emphasising a strong focus on individual contribution. Companies and other legal organisations cannot become full members, only (financially) supporting members without the right to vote in the general assembly. Employees of businesses may become members, but only on their own merit and in their own right. As a result of this the community is almost exclusively driven by volunteer contributors, with businesses invited to advisory roles.[49] In August 2008, the KDE approved a code of conduct as the first documented community behavioural guideline next to the bylaws. In October 2012, the community published the “KDE Manifesto”, which postulated norms like open governance, inclusivity and common ownership.[50]

The KDE community operates in a decentralised fashion with KDE e.V. as a central support organisation and has offices in Berlin. Multiple regional sub-communities exist at different levels of formalisation. Some like the ones in Spain, India and Latin America are represented by individual legal entities that are associated with, but not controlled by KDE e.V.

46 https://community.kde.org
47 https://dot.kde.org/2009/07/20/kde-reaches-1000000-commits-its-subversion-repository
48 The author of this study drafted the 2003 version of the bylaws in 2002.
49 Lydia Pintscher, ed. 20 Years of KDE: Past, Present and Future. KDE e.V., Aug. 2016.
50 https://manifesto.kde.org/

From the beginning, KDE e.V. was meant to support and represent the community. It was clarified in the 2002 general assembly that this excluded KDE e.V. from influencing the technical direction of the community product. The organisation is represented by the board and conducts an annual general meeting. In 2005, an attempt was made to establish formally recognised working groups that would coordinate with the board and be able to manage specialised budgets. Multiple working groups were established, but most ceased activities within a few years. No other formal structure has been defined within KDE e.V.[51]

From the start and in the initial stage, the KDE community considered itself a meritocracy. One of the first principles the initial group of about 10 contributors established was "(s)he who does the work decides", which postulates that even a community decision cannot force a technical direction on the person implementing it. The internal understanding was influenced by the publication of "The Cathedral and the Bazaar"[52], which many of the early contributors had read. The group choose a meritocratic, egalitarian approach to self-organisation, with the original founder acquiring the most impact. Communication differentiated into different channels, particularly mailing lists, to keep the distraction of ongoing debate away from contributors working on the product itself. There was general acceptance of the argument that an inner circle needs to exist to manage the project, first with the kde-private mailing list and to this day with the non-public KDE e.V. membership mailing list. Access to this inner circle is granted by the existing insiders.

Informal behavioural norms played a significant role within the KDE community. There is a strong resistance to any form of authority within the community that is not based on individual merit. The rule that KDE e.V. shall represent the project, but not influence technical direction is considered a fundamental constitutional principle that newcomers are introduced to very early on. The resistance to authority was also embodied in the "(s)he who does the work decides" norm. As a result, technical direction developed organically from the activities of the contributors. Voting and other forms of formal decision making are not highly appreciated and seen as measures of last resort. Votes are commonly conducted to accept new members into the organisation, and to elect board members and representatives to external committees or organisations. The importance of these principles contributed to the absence of organisational design in the late community stage.

One of the key debates that has never been concluded is whether or not KDE e.V. represents "the heart of the community" or is meant to be a body that complements the community without being a core part of it. The strategy of the organisation was from the beginning that active contributors should be members of the organisation, and therefore jointly own and manage funding and ownership of trademarks and other assets. To achieve this goal requires an organisation that is accepted by a meritocracy, which needs to aggregate the interests of many of the core contributors.

However, the more influential the organisation became, the more it came to represent the project overall, with the board growing into a sort of project leadership. This created a conflict due to the fundamental resistance to authority prevalent in the community. The community had not established processes capable of making decisions on questions of this constitutional nature. Decision making relied on the relevant stakeholders taking part in an extended elaboration with the goal of reaching consensus. For technical decisions, this approach served the community well. The community does not differentiate between product related technical decisions and the implementation of norms of social process. The decision-making process aimed at consensus proved to be less efficient for topics that affected all community members, where everybody is a stakeholder. Effectively, the formal organisation became very difficult to change, with the consensus driven process affording each individual member a de-facto veto.

51 Diomidis Spinellis and Georgios Gousios. Beautiful Architecture: Leading Thinkers Reveal the Hidden Beauty in Software Design. O'Reilly Media, Inc., Jan. 2009.

52 Eric S. Raymond. The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary (O'Reilly Linux). O'Reilly, Oct. 1999.

4.2.2. Decision making and conflict resolution

Because of the “(s)he who does the work decides” rule, a decision “manifested itself based on what ended up in the revision control system”. In the early and medium stages of community growth this approach served the community well. Possible differences would be settled by arguing for the cause until an agreement was found. The decision-making process relied on the organic coordination of a familiar, cohesive group with a common cultural understanding. Over the years, this attracted more contributors with the same traits, contributing to the common lack of diversity in FOSS communities: The contributors were predominantly young male software engineers. A growing community however requires increased specialisation and division of labour. Other community needs requiring skills like documentation, user experience design and community management were not met due to a lack of contributors as a result. Selective bias has been criticised as a hidden cost of meritocracies.[53] It can be argued that the community may not reach the full contributor potential because of it.

Participation in debates is open to all contributors, but the impact of their voice depends on their meritocratic status. This is not unusual in smaller collective action groups.[54] This attitude emphasises technical contributions over those in other fields. Minority opinions have a difficulty being heard. Because contributions are made voluntarily, there is only little participation of specialised and minority contributors.

Conflict resolution within the community is mainly absent. Except for appealing to the board of KDE e.V. as a general fallback option, there are no defined processes to escalate a conflict with the goal of settling it. Contributors are expected to sort things out amongst themselves. In 2008 the community working group was established, together with the code of conduct, with the aim to “maintain a friendly and welcoming KDE community, thereby ensuring KDE remains a great project enjoyed by all contributors and users”.[55]

The community working group moderates using a participative approach and is not equipped with any sanctioning instruments. The only possible measure to sanction misbehaviour is a suspension of a contributor’s accounts, either temporarily or, as an ultimate measure, permanently, a task performed by the system administration group. Account suspension constitutes a drastic measure, as it effectively removes the affected person from the community. It impacts contributors similarly to a citizen of a country being subjected to temporary exile or revoked citizenship or a church member being excommunicated. It also strips the person sanctioned from the means of communication needed to enable them to continue to be part of the discussion and defend their position.

Account suspensions are therefore issued only in a very few cases, and only after lengthy moderation failed to resolve the conflict. In some cases, this has delayed necessary responses to disruptive and abusive behaviour. Similar to the decision-making processes, the mainly informal conflict resolution mechanisms work sufficiently well in resolving product related debates with a small number of homogenous stakeholders with knowledge and a strong interest in the matter, and less well for issues relating to the community’s social process with a large number of stakeholders with only moderate interest. This indicates that both the decision-making and the conflict resolution mechanisms were established in the initial and medium stages of community development and have not evolved to be suited to the late stage where the social process grows more important than the product aspects.

While this description of the decision making and conflict resolution mechanisms within the community may appear as criticism, it mainly aims to describe the observable results and developments. The KDE community had good reasons to establish these processes founded in the

53 Daniel Bell. “On meritocracy and equality”. In: National Affairs 29 (1972), pp. 29–68.
54 Olson Mancur. The Logic of Collective Action: Public Goods and the Theory of Groups. Revised. Harvard economic studies, v. 124. Harvard University Press, Jan. 1965.
55 https://ev.kde.org/workinggroups/cwg.php

ethics of the community's early contributors. Well-defined formalised decision-making processes favour well-organised actors with the necessary resources to participate in these processes and who are expected to be businesses, not volunteer contributors. The absence of formal decision-making processes is seen as an emphasis of the role of the contributor of the product over others "that merely talk". Similarly, the apparent lack of conflict resolution mechanisms is also by design. One argument is that as long as the community has difficulties defining misbehaviour, it should not police it as that would result in arbitrariness. A second argument is that contributions that are disruptive to the overall technical direction of the community are important to innovation and should not be suppressed.[56]

The lack of formal definition of decision-making processes results in occasional over-the-top behaviour that consciously or subconsciously (even if with the intention of hearing all sides of an argument) prevent decisions from being made or from being implemented. Since there is no process that describes how discussions should be conducted, debates can become endless by bringing up a new arguments or points of view that need to be considered. It is not easy to distinguish between the contribution of an important argument to the debate and the deliberate raising of a tangential argument with the aim of derailing or prolonging it.

Bringing up tangential arguments that may or may not be critical to the conclusion happens often enough in the debates that it developed its own name, "bike-shedding",. This is named after the question of "which colour to paint the bike shed" when the debate is about "whether or not to build one".[57] The result is that debates may take much longer than the subject warrants, a topic will not receive the necessary attention, a decision may never be made, or a decision once made will not be implemented.

Occasionally, the community applies "lazy consensus" where the debate is settled by a contributor committing a solution that reflects what has been discussed, pre-empting further discussion. This happens more often with matters where stakes are difficult to define or controversial. In the KDE community, the KOffice versus Calligra discussion[58] or the decision to hire more staff, especially an executive director, are examples of discussions that dragged on for a very long time, sometimes years, before being concluded.

This pattern is even more difficult to manage because the individual contributor's reasons to prolong a debate may be sub-consciously self-serving but are rationalised towards the good of the community by the individual themselves, making them think they are acting in the best interest of the community. It gives individuals an instrument to abuse a participative debate culture that generally assumes good intentions. Instruments like time-boxing (limiting the period of debate by scheduling an executive decision or vote at the end) that are common in other collective action groups are not used in the KDE community because formal decisions through votes are not generally accepted. Interviewees suspected that the debate culture in the community was heavily influenced by the student lifestyle of the early contributors, dominated by non-structured arguments, a lack of any constitutional frame of reference and a lot of time for debate.

4.2.3. Community membership, roles and privileges

The KDE community implements an easily accessible "open door policy" to its core repositories, defined by the absence of any formal hurdles to gain access to the community infrastructure. Since all code and data is versioned, any change can be reverted, and there is no need for an approval process for contributor access. Only a small subset of the community infrastructure, for example the public-facing web sites, are kept under more restrictive control. Everybody who contributes to KDE

56 https://community.kde.org/Akademy/2013/ConflictResolution, there was no formal adoption of the suggestions.
57 Karl Fogel. Producing Open Source Software: How to Run a Successful Free Software Project. O'Reilly Media, Inc., Oct. 2005.
58 https://lwn.net/Articles/419822/

products or the community ("everybody on the mailing list") is considered a community member. To participate in the product related aspect this is all a contributor needs.

In the very early stages, a private mailing list was created for the more involved contributors, with an invite-only membership policy. It later evolved into the communication channel for the KDE e.V. members that is still kept private. At the AGM in 2012, an attempt was made to change this communication to make it public. It ended in the creation of the kde-community mailing list,[59] while the communication of the organisation is still private. There is privileged differentiation within the community regarding participation in the social aspect of community work, combined with significant barriers to entry like the invite-only principle. Contributors highly value being a community member, especially once becoming a part of the core team or KDE e.V.. Advancement to a role of formal community representative on the board or in external committees or foundations requires membership in KDE e.V. The KDE community is easily accessible for contributors to its product, but not as much to its social process.

4.2.4. Structural reforms and outlook

The formal organisation of the KDE community has been infrequently changed with the update of the bylaws, the introduction of the code of conduct, the publication of the manifest and the formation of the community working group, over the course of more than 20 years. Changes were incremental rather than disruptive, and retroactive in that they codified norms that the informal social process already had developed.

The informal organisation changed gradually but preserved "hacker culture". In the absence of formal structure, thought leaders have had a strong impact, with an emphasis on personality that is difficult to replace at a later stage.

If two contributors did not agree on which text editor KDE should ship, it would ship two text editors. There was no mechanism to influence technical decisions that affected the project as a whole and the users of the software. More importantly, there is no sanctioning mechanism to encourage activities that the community is interested in. The "(s)he who does the work decides" rule means that the user has to turn into a developer contributor to improve the software for her or his needs.

The community norms described have all been developed and adopted in the early stages of community growth. They worked well in the small to medium sized groups and did not change significantly in later stages when KDE decided to de- emphasise product development over being a community that creates FOSS products. An unresolved contradiction lies in the application of predominantly informal norms and ethics tailored towards a smaller coherent social group with uniform backgrounds and interests to the governance of a large, diverse organisation. The cultural foundation that the community codified in the vision and manifest is not implemented in its long-standing governance norms.

4.3. Wikimedia

4.3.1. Mission, foundation and history

The online encyclopedia Wikipedia was launched in 2001 with the vision of creating "a world in which every single human being can freely share in the sum of all knowledge". Unlike predecessor projects, it incorporated the idea that all content should be free with the same understanding as in free software. This vision was formulated in the early days of the project by the founder Jimmy Wales, and still remains largely unchanged.

59 https://mail.kde.org/mailman/listinfo/kde-community

Wikipedia is created by the global Wikimedia community. There is no central authority within the community that manages activities globally. Instead, regional sub-communities operate mostly autonomously, usually along language boundaries. It is therefore difficult to describe the governance norms of Wikipedia as a whole. In terms of contributors, community activities and funding raised, Wikipedia has a strong presence in central Europe, especially the German speaking countries. This study focuses on the community of contributors to the German language Wikipedia and the supportive organisation Wikimedia Deutschland e.V. in Germany. Initially, Wikipedia was perceived more as an idea, a broadly collaborative effort to make the knowledge of the world accessible to everybody and to enable them to participate as a user, author or community member.

The idea quickly turned into a global movement that attracted large numbers of contributors. In May 2017, about 120,000 participants actively contributed to the project. With over five million content pages and nearly 900 million edits, Wikipedia has successfully built the encyclopaedia it set out to create, surpassing commercial encyclopaedias by article count and number of readers. It gained a large user base in the process, serving in average about 7.8 billion pages per month. It globally ranks 5th in the list of most visited websites. With this initial success, some contributors shifted their focus towards building the very best encyclopaedia, with a focus towards quality over quantity. Others identified new fields and regions of knowledge that need to be captured and consider the task of collecting all relevant human knowledge far from completed. In any case, building the Wikipedia encyclopaedia is an ongoing, life-time scale undertaking. The question whether Wikipedia's mission makes it a project responsible for social change or for the concrete task or writing the best encyclopaedia in the world is still being discussed.[60]

4.3.2. Formal and informal organisational structure and conventions

There is only minimal formal organisation of the community of German-speaking Wikipedia authors. Wikimedia Deutschland e.V. represents the German language Wikipedia legally and provides community support. Similarly, the San Francisco based Wikimedia Foundation legally represents the global community and the English language Wikipedia and maintains a level of control over the regional organisations. These organisations however are not directly involved in coordinating or managing the work of Wikipedia authors and other individual contributors. Authors commonly focus on contributing knowledge in their own language and possibly to the English language Wikipedia, which is seen as the global fall-back. More than in other organisations, the contributor base is fluent, because it is possible to contribute, even anonymously, without much interaction with the organised community. Groups of regulars (*Stammtische*, in German language) meet occasionally to maintain cohesion between the work of the individual authors. Many participants expect regular contributors to attend physical meeting to gain recognition. Editorial boards have formed for specific subject matters like chemistry or religion.[61] Arbitration committees have been created in some countries (2007 in Germany) that assist in resolving conflicts between Wikipedia users. The arbitration committees do not interfere with regular contributor activity.[62] Beyond that, no formal structure exists that the authors turn to for coordination of their work Intentionally, no attempts are made to unify the processes of the regional sub-communities. Regional differences and decentralised self-coordination are considered key strengths of Wikipedia.

Wikimedia Deutschland e.V. develops software used by Wikipedia, especially MediaWiki, lobbies for open knowledge politically, invests into free learning and free educational resources, provides infrastructure and facilities for use by volunteers, and overall manages the organisation's and the community's legal and financial footprint. In 2016, it reached 50,000 individual supporting members,

60 A history of Wikipedia is available at https://en.wikipedia.org/wiki/History_of_Wikipedia. Details about the vision can be found at https://wikimediafoundation.org/wiki/Vision. Content page count, number of active users and other metrics are available on Wikipedia's statistics page: https://en.wikipedia.org/wiki/Special:Statistics. A user is considered active if she or he performed an action in the last 30 days.

61 https://de.wikipedia.org/wiki/Wikipedia:Redaktionen

62 https://en.wikipedia.org/wiki/Arbitration_Committee

2,000 voting members, and about 85 employed staff. It is led by the executive committee (*Präsidium*) of up to 9 members, which appoints the executive director. The activities of Wikimedia Deutschland e.V. are for the most part considered orthogonal or supportive to the work of the community of authors. Some participants in the interviews actively refused the notion that Wikimedia Deutschland e.V. is part of the German language Wikipedia community and consider both separate entities. Wikimedia Deutschland e.V. does not consider itself responsible for the activities of the community of authors. The relation between Wikimedia Deutschland e.V. and Wikimedia Foundation has been characterised as that of "a far-removed sovereign" that tributes are paid to.

The community organisational structure has been described as "profoundly informal". In particular, early community contributors or "generally important top-dogs" can be very influential, even without formal roles. It was pointed out in the interviews that this may impose significant barriers of entry for new authors. Formal and informal structures have diverged significantly. It can be assumed that the community is not in a position to perform an analysis of the state of the project and derive conclusions for organisational reform, which has been classified as "negligent" in the interviews.

4.3.3. Decision making and conflict resolution

The decision making and conflict resolution norms within the author community show strong similarities to those found in the other studies, underlining the assumption that governance norms evolve based on community composition.

Acknowledging that majority decisions cannot be enforced against volunteer contributors; they have been replaced with non-binding opinion polls (*Meinungsbilder*). All active contributors may initiate and participate in an opinion poll. There are strong opinions about opinion polls, with some arguing that contributors should participate in them, and other arguing against participation. An aversion against formal decision making is obvious. Similarly, the "rule to ignore all rules" encourages participates to apply agency to their actions.

Intra-community conflicts are managed along a well-documented staged process from de-escalation to appeals to a mediation committee (*Vermittlerausschuss*) with volunteer members and finally to an arbitration board (*Schiedsgericht*) with members that are elected today by a qualified majority. Decisions of the arbitration board are considered binding within the community. Recommendations like remaining level-headed and assuming good intentions help to maintain a collaborative spirit, as do more formal guides like the "Wikiquette". While there are instances of "edit wars" or members acting under fake accounts ("sock puppets"), the conflict resolution process is mostly accepted and effective. These processes represent a mature understanding of the role of decision making, conflict resolution and of volunteer community dynamics. None of them involve Wikimedia Deutschland e.V..

4.3.4. Community membership, roles and privileges

Everybody who productively contributes to Wikipedia is considered a community member. Since anonymous contributions are allowed, contributors transition from loosely associated anonymous authors to registered authors known by a screen name and then may acquire additional roles like administrators. Elected *Bürokraten* (bureaucrats) manage administrator status. A number of additional roles exist that partially map to technical permissions in the operation of Wikipedia, like rolling back changes or inspecting contribution metadata. There is consensus that all contributors should be considered equals, taken seriously and valued based on merit. Even though being admin is foremost a technical task that allows to change other contributors' content, it is also implicitly a social role that needs backing by the community and therefore a strong standing or merit for the person acting as admin. Eligibility to vote is based on a minimum number of recent contributions, and since the bar is set rather low, has become a requirement for effective participation in discussions. Without it, an individual "would not be taken seriously". Long-term contributors who shifted their focus towards activities other than being authors sometimes struggle with that or produce edits to maintain their status. Social status within the community is closely related to contributions either of quality content or of the software used to run Wikipedia. Contributors to auxiliary functions like conference organisation or design are "not well known".

These status groups or roles represent contributor functions with a strong product focus - they are measured against their impact on the quality of the encyclopaedia. A remarkable disconnect was mentioned in the interviews between the legal entity Wikimedia Deutschland e.V. representing the German speaking Wikipedia, and the community of authors. One described the role of Wikimedia Deutschland e.V. as "collecting donations, being on TV, and attending galas, based on the work of the community". Multiple interviewees mentioned that being a member of Wikimedia Deutschland e.V. was perceived in the past as a negative factor with regard to contributor merit within the community and is now considered "acceptably eccentric". The role of Wikimedia Deutschland e.V. members within the community is largely irrelevant, except for a small number of contributors that try to participate in both organisations but find it time consuming and difficult. Wikimedia Deutschland e.V. has been repeatedly criticised for being disengaged from the community and not supporting it enough. Interviewees expressed that they believe the perceived under-performance of Wikimedia Deutschland e.V. is rooted in the lacking integration with the community, and that they "are happy if Wikimedia Deutschland e.V. is at least not breaking anything". Wikimedia Deutschland e.V. keeps authority of the budget and spends a significant share of the budget on non-product related activities.

4.3.5. Structural reforms and outlook

As with the other case studies, the formal organisation within the German Wikimedia author community has rarely changed. There is no structured process of organisational design review. In the past ten years, the adoption of the review principle, the arbitration court and the introduction of the visual editor are perceived as the major changes. Interviewees described the overall constitution of the community as rather "hostile to change".

Wikimedia Deutschland e.V. is aware of the rift between the author community and the formal organisation. It attempts to integrate the community through a collaborative planning and budgeting process and other activities. Decreasing author numbers create the necessity to act upon a perceived pent-up need for organisational reform, which is reflected in the annual plans for 2016, 2017 and 2018. Attracting and retaining volunteer contributors has been accepted as one of three key fields of action. However, less than ten percent of the overall revenue from donations and membership fees is allocated directly towards that goal. Wikimedia Deutschland e.V. positions itself as an organisation with the primary goal to foster Wikimedia projects.[63] There is a profound feeling within the community of authors that Wikimedia Deutschland e.V. made itself independent and unaccountable. When asked what would need to change, one interviewee suggested that community members need to "get together" and re-take control of their project.

5. Observations

For the most part, the contributors agree on what they expect from their communities: They want to engage in a community of "makers". Amongst their peers, they wish to have equal opportunity to contribute. They understand the need for community management but want their communities to

63 "Wikimedia Deutschland [ist] im Hinblick auf die Wikimedia-Projekte daher als "Förderverein" zu verstehen." (https://meta.wikimedia.org/wiki/Wikimedia_Deutschland/Präsidiumshandbuch)

remain focused on being ambitious, productive meritocracies. They believe that there is strong solidarity between the members of their communities, and that “them-versus-us” conflicts between the communities and their leadership or the makers and the community builders are mainly absent. Still, the communities exhibit similar symptoms of distress: They have trouble growing their contributors and contribution count in a sustainable way, have difficulties implementing organisational change and get stuck making important decisions, resolving inner-community conflicts or enforcing the values of their social groups.

It is obvious that the contributors think highly of their communities. There have been no indications of any malicious intent by influential participants or abuse of the communities for their own advantage. Conflicts within late stage FOSS communities are more likely to reflect difficulties volunteer contributors have in collaboratively developing their organisations and maintaining control over their destiny as they grow to be large groups.

5.1. Formal and informal organisational structure and conventions

Formal organisation is not the first thing participants have in mind when starting a FOSS initiative. The groups are initially small and do not possess assets or liabilities that require an independent formal organisation. KDE and Wikimedia Deutschland e.V. added a legal entity that represents their community after their projects started. The founders of FSFE on the other hand were aware that their success depended on a strong, independent organisation, and started off with a carefully designed organisation that anticipated attempts of hostile take-overs and the creation of regional subsidiaries.

None of the three organisations implemented a systematic effort to periodically review and reform their formal organisation. Changes to the organisational structure have been very rare, resulting in a growing disconnect between the community’s production processes and their governance related activities.

FSFE did not succeed in establishing thriving, decentralised, independent regional sub-organisations, and concentrated its activities at the Berlin head office.

The KDE community continuously restricted the mission of KDE e.V. to administrative support and resisted the delegation of authority to elected representatives. This led to long-standing contributors questioning its usefulness and contributed to an emerging culture of bike-shedding and indecision. The organisational structure of KDE e.V. was not changed even as the KDE community changed from a single product to an umbrella community. Eventually, KDE e.V.’s main role became to organise the annual KDE Akademy[64] conference and to provide funding for contributor meetings. For a period of time, the KDE community became infamously known for its lack of coherence and decision making.

The biggest rift between product development and formal organisation seen in this study is exhibited by the German language Wikipedia community. Despite the minor differences FSFE and KDE have with their formal organisations, they are still seen as an integral part of the community. Some members of the community of Wikipedia authors however wish that Wikimedia Deutschland e.V. would “not interfere with their work”. Perhaps because being a Wikipedia author is explicitly not considered a selection criterion for employment at Wikimedia Deutschland e.V. the community of volunteer authors and the formal organisation that bears the community’s name have diverged.

The performance of the support organisations in this study is not linked to the efforts invested into the original organisational design. Instead, given the absence of a systematic review and reform process, the organisations’ ability to serve their communities deteriorated as they went from the

64 https://akademy.kde.org

initial to the medium to the late stage. The aversion of the contributors against authority and "bureaucracy" put the need for reform in question and reinforced this trend. It is not the initial design that counts, as the organisations need to continuously adapt and improve the performance at which they support their communities.

All three organisations rely heavily on informal organisational structure. There is strong agreement between the interviewees that the documented formal structure is not implemented in reality, and that today the organisational structure of the communities is mainly implicit, well-understood only by early community members, and not well-documented for newcomers. Only a few fully know and understand the existing formal rules as they stand today. The divergence of formal and informal organisation and the lack of supportive performance of the community organisations is not currently perceived as a relevant problem. The resulting effects, enforced by lack of positive competitive selection of community leadership or inhibited acquisition of new contributors, are detrimental to long-term community growth and success.

5.2. Decision making and conflict resolution

It is commonly part of the spirit of a FOSS community that decisions should be made by consensus, that authority and hierarchy should be avoided, and that there should be minimal to no policing of contributor activity. These are all positive, defining aspects that are important to contributors. But do they match reality when compared to the decision-making processes and conflict resolution mechanisms of late stage communities?

The results from the interviews strongly suggest that all three communities make use of very few defined decision making processes, do not routinely apply instruments for shaping debates, experience extended bike-shedding and indecision regarding issues that are considered important, and that influential individuals – often project founders or early contributors – wield soft and hard vetoes over community decisions.

Most day-to-day decisions are made at the level of subgroups that focus on particular aspects of the community product. In these relatively small groups, informal decision making still succeeds. It is possible to understand the likely outcomes of the decision, and there is a joint sense of responsibility for that result. There is also no need for an appeal mechanism. If the outcome of the decision is not what was expected, the group again jointly decides on a new course of action. These are the decision-making mechanisms the communities developed in the early stages and that served them well.

Late stage communities also need to make more complex decisions, like hiring an executive director, organising a global conference or redefining the overall community vision and mission. These may involve trade-offs of resource allocation between subgroups or competing goals. The community as a whole is a stakeholder in these decisions. The outcomes of the decisions may be harder to predict, and unlike most technical decisions difficult to reverse. Undefined and informal decision-making processes, a lack of routes of appeals, and an excessive debate culture that may prevent decisions from being made have a detrimental effect on contributor motivation and pose a significant barrier to entry into higher level community functions. Early contributors stay in community leadership roles too long, at the expense of later contributors not assuming leadership roles even if their merit within the community would warrant it. The auto-organised decision-making mechanism of the subgroups fails when applied to higher level large group decisions.

Authority is commonly assigned to specific community functions, like the president in the case of FSFE, or the board in KDE e.V. There are no checks and balances to decisions made by these functions. Even if it is known to some participants that a way to question a decision of the president is to submit an item to the agenda of the next general assembly, this is far from obvious to the wider community, and also not communicated. There is little understanding that for every community function with authority a check needs to be implemented to allow oversight and for decisions to be appealed. In volunteer-driven communities with self-referential authority, this means that unresolvable issues eventually will escalate to a community all-hands decision. In turn, this requires a mechanism for community votes. Communities have a well-founded aversion to majority decisions considering that all contributors participate in the communities voluntarily. However, as a mechanism of last resort, no better alternative has been presented. Every decision should be appealable. Many of the long-term contributors interviewed in this study today acknowledge the need for well-defined decision-making processes, even though it took a long time for them to reflect on and change their initial preference for auto-organisation at all levels.

With unclear authority, it is difficult to apply instruments to shape debates so that a decision can be achieved in a reasonable time frame. Discussion drag on "until nobody has any energy left to disagree". A "fulsome optimism regarding the decisiveness of a large group" can be observed. Discussions are kept alive by influential contributors to avoid their conclusion with an unwanted decision. Shaping debates does not necessarily require voting mechanisms. Time boxing by asking that a consensus be reached after a specified discussion period and announcing a formal vote otherwise is one option. Relying on rough consensus combined with a clear procedure of appeal is another. The KDE community had good experiences with a "debate manager", a contributor that voluntarily steps up as a moderator and drives the debate to a conclusion. It could be expected that by combining decision making and appeal processes more clearly and organising debates in a more result-oriented fashion, the tendencies towards bike-shedding and indecision that communities exhibit can be overcome.

5.3. Community membership, roles and privileges

The communities apply a broad definition of what makes a community member. "Everybody on the mailing list" who actively participates is considered to be one. Those who contribute more, over an extended period of time, begin to form a loosely defined "core team" early in the process, which also separates those who "merely talk" from those "doing the work". Formal membership in the support organisation forms another community rank. Being appointed to a board or elected leadership position is another one. This suggests a hierarchy of influence that may be misleading, as advancing through the community ranks does not necessarily happen on a straight career path. The differentiation between product contributions and community management may lead to contributors gaining leadership positions that never contributed to the community product. Authority is also gained ad-hoc by individual contributors self-identifying with the initiative to manage a debate, or a community process like writing the manifesto. It can be observed that once contributors reach a board or elected representative level position, they rarely ever go back to being regular contributors. This indicates that such positions do form a sort of end-of-career achievement. Community rank is considered significant in that individuals would, for example, list their community achievements in their CV.

Contributors advance through the community rank meritocracy based on their contributions. Not all contributions are valued the same. Contributors to the core product, founders and individuals "rich on time" advance through the meritocracy more easily. Typically, contributors gain more merit when contributing directly to the community product, as opposed to covering support functions as in helping with administration or event management. Even auxiliary product contributions like the work done by designers and documentation authors are less likely to be appreciated. This may inhibit effective specialisation as the different "professions" within the community carry different merit. The inherent contradiction of the "who does the work decides" rule applied by many communities is that in an advanced community it is almost impossible to identify which specialised task is more important and who does the work.

The phase during which the contributor joins a project also affects the opportunities to advance through its meritocracy. Project founders and early contributors often remain in an influential position over a long time. Sometimes they evolve to be "luminaries" or "top dogs" that carry strong influence over community processes and are involved in many community decisions even without a formal role. Long-standing and early contributors aggregated merit that enables them to influence the community as a whole. This poses a difficult barrier for newcomers to become contributors, and even more for existing contributors to advance within the community status groups. The communities are aware of barriers to entry and work to keep them low, It seems however that the barriers are higher, not lower, for advancement and individual personal development *within* the community.

Many of the interviewed project founders and early contributors who rose to community leadership functions in the early and medium stages of the project emphasise that their motivation was to help the project, not to further their personal reputation. Some say that they were willing to do the work no other contributor wanted to do. Others stress that the perception of the president's position is of much higher value to others than to them. Their expectation towards other community leaders is that they would also mainly work towards the interests of the community, not their own benefit. Some interviewees admitted that other contributors might regard the group of founders and early contributors as a "round table" that is a bit out of touch with the rest of the community. The modesty expressed by the founders and early community leaders is convincing in the early and medium stages of community growth. For late stage communities, it must be assumed that the prestige and remuneration for serving in a community leadership role becomes an attraction in itself. Late stage communities will then require a system of checks and balances to maintain control over community management, which was unnecessary and therefore not established in the early and medium stages.

Similar to organisational structure there is an implicitness in the community status groups, roles and privileges. The self-referential authority within the community is well-understood by the founders and early contributors. One interviewee said, "I set the rules once, I can do it again." This freedom to question rules and apply norms where they are applicable and ignore or bend them otherwise is second nature to old-timers. It is explicitly communicated, as in "if a rule prevents you from improving or maintaining Wikipedia, ignore it", but difficult to grasp for newcomers. Rules solidify, sometimes unwantedly, and are rarely ever changed for a late stage community. The open doors policy that the communities are proud of deteriorates in the late stage. Long-standing administrators expressed worries that "some of us have lost the trust that newcomers will do good things".

5.4. Structural reforms and organisational design

In all three cases, the explicit and implicit organisational structure emerged in the early and mid-stages of community development. While the initial structure developed implicitly, the communities did create well-working formal supportive organisations that originally served their purpose well. However, they did not implement a systematic and periodic review process which ensures that implicit and explicit structure and processes do not diverge too much, and that the formal organisation stays focused on the mission that the community created it for. They also implemented partially insufficient checks and balances to enforce accountability of these organisations towards their contributor base. Through membership open to all active contributors and direct as well as competitive elections of community representatives by the members, KDE e.V. remained most effective and accountable to the community of the three cases.

With the removal of the elected fellowship representatives and the position of the executive director, FSFE grew less accountable in 2018. While it does still represent the ideals of software freedom and aims to speak for the wider Open Source community, it gains few new contributors.

Of the three cases, Wikimedia Deutschland e.V. developed to be most removed from its original purpose of serving the German language Wikipedia community. While it drives fundraisers and shares the name of the community project, most of its activities and most of its budget do not directly support the community of authors. While all its activities are charitable and contribute to the cause of free knowledge, almost all they share with the community of authors is the name. It is reasonable to assume that this fact contributes to the declining number of authors, as potential contributors realise that their work is being used to raise funds for mostly unrelated activities and a large body of staff.

All communities exhibit an aversion against administrative processes or “bureaucracy”, resulting in an apparent lack of momentum towards active organisational change. Interviewees from all three communities mentioned that formalising Structure and documenting, decision making and conflict resolution processes to a necessary extent helps to maintain the freedom to participate and joy of contribution. They face the challenge of preserving “hacker culture” while at the same time enabling large numbers of contributors to collaborate successfully.

Based on the lack of organisational design, community processes and structures are largely implicit and there are no well-defined rules of appeal. The German-speaking community of Wikipedia authors provides a positive example of well-working auto-organisation. However, these processes are independent of Wikimedia Deutschland e.V. as their support organisation, indicating that the resulting lack of accountability offers opportunities for self-serving behaviour.

This raises the question of how communities can ensure that their structure and processes evolve so that they continue to fulfil their mission of supporting their contributors. Where competition keeps businesses aligned with their purpose and elections align the actions of politicians with the interests of the population, FOSS communities depend on voluntary participation to raise contributions. This postulates the number of independent contributing entities and the number of contributions raised as key metrics for community health. Implicit and explicit community structure and processes should primarily aim to support these goals. Community activities, also by the support organisations, should be assessed based on how they contribute to these goals. From the budget a community is raising, every Euro that is spent on activities that do not contribute to these goals reduces the number of attracted contributors and through that the potential impact and success of the community. The self-referential purpose of FOSS communities means that all functions of the community need to be accountable to the base of its active contributors. In turn, a community can only represent those that by way of actively contributing acquire an equal voice in decision making and conflict resolution processes.

6. Summary

This study started out from the observation that FOSS communities struggle to maintain growth once they reach a large number of contributors. It could be observed that the growth phases the communities proceed through can be grouped into an initial stage with ad-hoc coordination and an equivalence of individual and group goals, a middle stage of growth with consensus-focused auto-organisation and a late stage with more profound functional differentiation and formal structure.

Businesses, individual volunteers and staff members participate for different sets of reasons. The concept of community composition refers to the mix of volunteers, businesses and staff that engage in a community. Assuming that, all things being equal, governance norms develop depending on community composition. The study analysed three primarily volunteer driven communities to provide an insight into their governance and to identify commonalities between them even though they create vastly different products. Based on the principle of voluntary participation, the purpose of communities is defined in a self-referential manner: The community serves the interests of the contributors that form it, with no outside authority except the law. This means communities need to solve the constitution problem to define who has a voice and to establish structure as well as decision making and conflict resolution processes, based on voluntary participation.

Community governance is shaped by the mindset of the contributors. Individual volunteers are primarily intrinsically motivated individuals, which is reflected in the expectations they expressed in the interviews: motivated to participate in a community of makers, to experience equality of opportunity among their peers, to find a balance between makers and community builders, to become a part of an ambitious, productive meritocracy and to see their own ethical principles represented in the community governance norms. If these expectations are fulfilled, they develop increasing loyalty towards the community.

The case studies reflected the concepts of membership, the formal and informal organisational structure and the decision making and conflict resolution processes of the communities against these expectations. They indicate that while there is a close or close enough match in the initial and middle stages, in particular the formal organisations that have been created to support the work of the community show a tendency of distancing themselves from the community goals. The combination of solidified implicit norms and more closed-up organisations creates barriers to entry for newcomers and reduces the number of long-term, loyal contributors the community is able to attract in the late stage. While it remains relatively easy to contribute to the community product, it becomes increasingly hard to gain access to influential formal roles and positions.

The gap between makers and community builders grows with size of the community. Independent of the effort invested in setting up the original support structure, the formal organisations partly disconnect from their communities. This seems to be caused by the absence of a regular review process based on checks and balances built into community governance, resulting in a lack of accountability of the support organisations towards their communities.

Volunteer contributors exhibit aversion to authority and formal decision making. At the same time, they jointly form the highest authority within their community. A possible conclusion is that community decision making processes should be well-defined, and that the highest level of escalation should be the community as a whole. Conflict resolution mechanisms should mirror the decision-making processes.

The communities investigated in this study partially lacked instruments to ensure that their structure and processes supported the overall community goals. Similar to elections in politics and supervisory boards representing investor interests in enterprises, communities need to re-align decision making power and require accountability to remain volunteer driven in a successful transition into the late stage of community growth.

About the author

Mirko Boehm is a Free and Open Source Software contributor, primarily as a developer and speaker. He is the founder of the Quartermaster project, and has been a contributor to major Open Source projects like the KDE Desktop since 1997, including several years on the KDE e.V. board. He is a visiting lecturer and researcher on Free and Open Source Software at the Technical University of Berlin, part of the FSFE Germany team and a Qt-certified specialist and trainer. Mirko Boehm has a wide range of experience as an entrepreneur, corporate manager, software developer and German Air Force officer.

The Open Invention Network protects the Open Source ecosystem by acquiring patents and licensing them royalty free to all participants. As director for the linux system definition, Mirko Boehm is responsible for the technical scope that defines the field of use of the patent non-aggression agreements.

As a co-founder of Endocode, an employee-owned, shareholder company based in Berlin, Germany providing professional IT services with a focus on Open Source technologies, Mirko Boehm specialises in consulting to and mentoring startups and medium to large businesses. His areas of expertise include complex software development endeavours, the use of Open Source products and methods in organisations, and technology related issues of business strategy and intellectual property.

He lives with his wife, two kids and two cats in Berlin.

Open Source Compliance and Goldilocks: Too Little, Too Much, Just Right

Robert Marion [(a)]

(a) Sr. Open Source Compliance Analyst at Citrix

DOI: 10.5033/jolts.v10i1.138

Abstract
Although open source compliance can be tricky and require resources it is not inherently necessary to employ dedicated personnel and enterprise solutions to accomplish a reasonable degree of best practices. This is particularly in situations where any resource allocation will compete for the resources required to actually create a product.[†]

Keywords
Law; information technology; Free and Open Source Software; licensing; compliance

Software developers depend on open source code. They pull it from GitHub, SourceForge, the Python Package Index (PyPI) and many other sources. There are Open Source advocates, who believe all software should be open source and there are professional organizations like OpenChain[1], the Open Source Initiative[2] and the Free Software Foundation[3] which assist in defining open source compliance programs. There are also foundations that produce, host and represent the Open Source community such as the Apache Software Foundation.

In contrast, businesses often substantially depend on their proprietary code to build products and generate revenue. They have a need to demonstrate that they know what is in their code because a software product is an assembly of many different components and these components may be open source projects (or commercial) and will be comprised of multiple authors, copyrights, and licenses. The many parts of a software product are often obscured by the ease with which one can incorporate code that does not originate with the product's developers. Entire software architectures can be built around a third-party framework before consideration is given to whether its license is compatible with proprietary code. One example is that in the case of the General Public License (GPL), other open source licenses may be non-compatible with the GPL's terms. A license establishes both rights and responsibilities and a failure to understand the responsibilities of a license can lead to litigation, monetary losses, missed deadlines and the need to halt shipment of a product. A growing awareness of licensing obligations with the use of third-party open source means that many companies now have introduced mechanisms for compliance. However, with a startup company, the compliance

† The views and opinions expressed in this article are those of the author and do not necessarily reflect the policies, opinions or position of Citrix Systems.
1 https://www.openchainproject.org
2 https://opensource.org/
3 https://www.fsf.org/

program may not exist at all or it may be as light as asking the developers to be wary of using code that is “adversely” licensed. Even with larger companies there may only be a single person responsible, often a lawyer, or a department of persons using enterprise software tools to prevent the inclusion or detect existing code that has licenses that are not considered proprietary friendly.

This article will consider the notion of there being a waterline for a right amount of compliance. A tech startup will not, nor should they have a multi-person compliance department any more than it would make sense for a company with hundreds or thousands of developers to have a single person who may have other duties running a compliance program. In all cases, there is a need to balance licensing compliance with the amount of resources available to a business.

No Free Beer for You

How Free Open Source Software (FOSS) is defined can be somewhat contentious with various organizations evangelizing their perspective of what is reasonable and correct. Some entities and groups share a belief that software should be freely shared for the benefit of all humankind, or at least, all programmer-kind. It is an altruistic philosophy, a democratic ideal that software products should be thought of as ideas that do not belong to a single class of people. The Free Software Foundation defines free open source software as having the freedoms to use, run, modify and redistribute a program. We “should think of free as in free speech, not as in free beer”.[4] This belief is often at odds with the imperatives of the business world, which seeks to monetize everything possible to provide returns to their stakeholders. Yet somewhat ironically, commercial enterprises benefit most from open source.

Open Source licenses mostly fall into one of the following buckets: Public Domain (not actually a license, but a dedication), Attribution Style (such as the MIT or BSD), weak copyleft (LGPL), and strong copyleft (GPL or AGPL). When it comes to compliance, the copyleft licenses are less “proprietary friendly” than the others and some compliance programs only focus on those license types via various degrees of exclusion. Increasingly, companies are concerning themselves with the greater task of finding all the open source in their products and creating a Bill of Materials. An example is that your mobile phone – whether from Apple or an Android provider - will contain a list of the open source third party software present. Companies are also placing their catalogues of open source components online. For example, you can see all the third party open source that goes into Ford Motor Company’s infotainment system at https://corporate.ford.com/legal/ford-open-source.html. It is an extensive list. By creating this information and placing it online, Ford is telling its customers that it is compliant and that it has a process in place. That process is designed to instil confidence on the part of Ford vendors and customers that they themselves will know what is in their products (if, for example, they were to incorporate Ford software). Of equal importance is the capability of knowing whether a third party component contains software vulnerabilities. The famous Equifax hack resulted in millions of credentials being lost, costly legal repercussions, and the firing of CEO Rick Smith. Could this particular breach have been circumvented? Yes. Equifax was using a third party component with a known and published vulnerability. Not knowing what they had in their software cost them millions of dollars and a seriously tarnished reputation.

Why should anyone care? After all, when I buy/license software I am trusting someone else to take care of compliance. So Ford, or the Acme Anvil Company or any other enterprise from which I purchase (actually, license) software is responsible for knowing what they have provided and what their legal obligations may be. Yet unless my engagement with the software involves no further distribution this stance does not hold true, as copyright licenses such as those found in software apply on any instance of distribution. If you inherit a problem, it can become your problem if adequate understanding is not present. Invariably, whenever a licensing snafu becomes public, the responsible

4 https://www.gnu.org/philosophy/free-sw.en.html

company will blame an outside consultant, contractor … whomever, but this stance in no way mitigates their responsibility to make sure what they inherited and subsequently passed on through the supply chain is correctly and sufficiently compliant. Hence the need for a Bill of Materials (BoM).

Compliance must address one issue: licensing obligations must be met. If your goal is to conform with licensing terms, you must give proper attribution where it is required. The BSD and MIT are examples of attribution style licenses. If a license is copyleft, then the user's obligations are greatly increased and so, by the way, is the likelihood of litigation. It is my personal contention that fear of litigation should not be the motivation behind meeting one's licensing obligations. There are two better reasons. First, obeying FOSS licensing terms is the right thing to do from an ethical perspective. The Zlib license expresses this in plain English: "you must not claim that you wrote the original software." That should be easy enough to understand. Additionally, companies and persons who are reputable should have an easier time selling their products if they are perceived as being fair and ethical and software-community friendly. As mentioned, the ability to produce a BoM may be a necessity for management, and it equally can be simply appreciated by customers as an example of care and consideration for both acknowledged authorship and legal correctness.

Why is Compliance Difficult?

Both the proliferation of open source licenses and terms of certain licenses have made compliance a non-trivial matter. As of this writing, there are over eighty open source licenses listed by the Open Source Initiative. For a quick comparison of many of these licenses in table format, see the Wikipedia article 'Comparison of free and open-source software licenses.'[5]

The GPL v2.0 license has 2,965 words. The GPL v3.0 clocks in at a hefty 5,660 words. Copyleft licenses evoke ideas such as derivative works, methods of compilation, distribution, aggregate works, etc. Some licenses that are commonly applied to software appear to be designed for other works of creative good such as the Creative Commons family of licenses and their origin in literature, imagery and similar works. Sometimes, knowing what licensing terms are applicable and how their obligations can be met is a sticky situation. Software engineers are not expected to be lawyers or licensing experts – but they do need to be familiar with the basic concepts of licensing. Furthermore, it is common to see upward of twenty different types of FOSS licenses in a single product. One product I recently looked at had over fifty different licenses or license combinations. Those are a lot of terms with which one must be familiar! This is not a simple matter if the correct approach is not used.

Complicating the issue of compliance is that software engineers can be notoriously independent minded given the nature of their work. Even if they are aware of licensing issues, they cannot be inherently trusted to be diligent given the practical difficulties imposed by writing software that works and delivering it on time. They may simply have other priorities, a situation that can be understood without judgement, but with the observation that it exists in production environments.

With all that said, one of the most difficult challenges with meeting FOSS obligations is knowing what is in your code. Not too long ago, it would be possible to pull the code from a repository and analyse it for third party materials. This has become much more challenging with the growing popularity of build systems that assemble code at build time (or even run time). For example, why have a hard copy of jQuery in your company's repository if you can fetch the most up-to-date version of what you need from an online CDN (Content Delivery Network)? Software as a Service such as The Cloud, and containers such as Docker, make knowing what goes into a build even more

5 https://en.wikipedia.org/wiki/Comparison_of_free_and_open-source_software_licenses

difficult. As alluded above, modern build processes may define what code will go into a program, but that code does not necessarily need to reside on your servers. This is becoming the norm rather than the exception and it creates a further challenge in knowing what third party open source you are building into your product.

Psst … You Wanna Buy a Company?

Mergers and acquisitions (M&A's) are an important part of the tech ecosphere. When larger technology companies wish to acquire another company, both sides must perform their due diligence. A business that is acquiring a software company will want to know that the software assets they are purchasing can be monetized. A company that is being bought may need to prove that the software they built has commercial value. The target company may need to prove that the intellectual property they claim is theirs actually belongs to them or is freely available under a proprietary-friendly license. If the software assets are unknown (the company cannot identify what goes into their build) or is built on non-proprietary-friendly licensed software, then it is possible for a deal to fail or the acquisition terms may be re-evaluated. Neither option represents a favourable outcome with the exceptional of marginal benefits for the purchaser in revising acquisition costs down.

It follows that an acquiring company will almost always seek the assistance of a law firm that specializes in these types of deals. When it comes to inspecting code for licensing issues … well, there are companies that specialize in that also.

What is the Right Amount of Compliance

The right amount of compliance not only depends on the size of the company, but the type of software product you are developing and how it is distributed. Projects may be entirely open source, may be proprietary but have open source APIs or may be what is now referred to as open-core. Consumer electronics and mobile apps are a much higher risk than web services because of the distribution model of each. That is to say, in the cases of web services, it is likely that third party components sitting in the cloud are often not considered to be distributed with the exception of code under the Affero GPL, which provides some mechanisms to address this use-case.

Open-core is a new name for something that has been going on for years. It is a combination of open and closed source, with the chief idea being that you can get some community contribution and acceptance and allow for some usage of a product without such users needing to purchase the software except in certain situations. These situations can often be defined as the full enterprise experience, various additions which will not in themselves be free or open source. This approach appears to be achieving a greater degree of popularity than it once had and, now that it has a name, also seems to have gained greater legitimacy. From an open source compliance perspective, open-core can be difficult to manage, and the cut-off point between the open code and the closed additions can cause friction.

The Cloud has changed what it means to deliver software. Not too long ago, purchasing software meant getting a media disk or downloading a product onto your server. That model of software delivery is becoming less relevant given how software products are delivered today, with The Cloud or SaaS rising dramatically and on-premise solutions are becoming less popular. This is important to open source licensing because most licenses were written before cloud computing existed. It is legitimate to believe that a GPL licensed software library may be used in the cloud and that does not constitute delivery and therefore the terms of the GPL license are not relevant to that situation. The Affero GPL (AGPL) license addresses that situation, but it has not been widely adopted, and certainly has not found favor with any significant number of companies in this space.

Let’s pull back to the assertion from earlier that open source compliance is not easy. Years ago, when I was developing software for a large company, one of my colleagues handed me a clipboard and told me that, if I ever used any GPLd software, I should log it on the clipboard. I nodded my head knowingly and he left. I had no idea what the GPL was and there was little danger of anything being recorded on that clipboard. Fortunately, as I recall, I wrote almost everything by hand. Today, there is a word for people who write everything by hand: unemployed.

So how much compliance is required? That is sort of a trick question. If you and your company are developing software that will be distributed, then you and your company are expected to obey all intellectual property laws and be aware of the licenses of all third party materials incorporated into your software. In a very real sense, I have never seen 100% compliance and mostly there are no consequences. However, M&As have been aborted when it was discovered through an IP audit that the chief product being acquired was really a derivative work of GPL licensed software. One company, for which I was contracting, had been sued in the past for not complying with the GPL and I have seen one case where a company purchased a competing company’s product so that it could analyze it to determine if they were violating any software licenses (especially the GPL). I call this “weaponizing” the GPL. These things do not happen often, but they do happen. For example, former Linux developer Patrick McHardy sued Geniatech and over thirty other companies in German courts over purported violations of the GPL. McHardy was unsuccessful in the Geniatech case, but his efforts overwhelmingly appear to be financial gain and not the safe-guarding of open source principles.[6]

Compliance is a Process

Some companies take a “ticking the box” approach. They may purchase some expensive software that promises to analyse their code and produce a report of all the third party components along with their licenses. I have evaluated many of these products and none of them can really accomplish the goal of sufficiently discovering third party materials. Why? There are a variety of reasons. Much of it has to do with how modern software is built. It is (or was) common for jQuery to be pulled in at run time by grabbing code from a “trusted” repo when it is needed. Sometimes code is pulled in programmatically at run time. Another example is Docker. Docker provides a platform-as-a-service and it also complicates life. Source code can be wholly taken from a third party project and it can even retain attribution, but it may be undetectable by enterprise scan tools. Such tools tend to do a great job at detecting unmodified binaries because they have a unique hash signature that can identify them. However, a simple JavaScript code section or snippet may go undetected.

It follows that simply purchasing a product and running scans is unlikely to produce a desired level of compliance and that a coherent process is instead required. From what I have seen, OpenChain has done an incredible job of defining those process inflection points, policy needs, and training approaches. For small companies, the overhead, time and effort required to implement those processes may be unaffordable in terms of both time and money. Instead, they may wish to do the following: train their engineers in open source licensing concepts. This may consist of a brief course given online or in-person. A trained engineer will know, most importantly, to record the license of a third party component. They may seek permission informally or formally and make an intelligent decision as to whether a given library poses an intellectual property risk. A small company that produces a non-trivial amount of code may also want to have an annual IP checkup by a consulting company.

Whether a company is large or small, the first step to compliance is training. A well-trained engineering staff, including managers, will ask the important questions before they become problems. Catching a problem after third party materials are introduced into your codebase may

6 https://www.zdnet.com/article/linux-beats-internal-legal-threat/

require costly refactoring and testing. So, the first step, whether a company is small or large, is to educate all involved in the production of a software product in basic license compliance. The Linux Foundation has a great course[7] that is accessible to beginners and also useful to those who are already familiar with FOSS. I also highly recommend the following video by IP lawyer Heather Meeker.[8] If training is not mandatory, then it may not be effective. Training is the very foundation of compliance.

Formal documented procedures, policies, and resources must be a part every company's FOSS compliance program. Policies should include more than how engineers should handle copyleft type licenses, but attribution style licenses, commercial licenses, situations where a license is unknown or does not exist. Policies must also consider inbound and outbound materials. Of equal importance is knowing what the workflow should be when licensed material, which is incompatible with a company's intellectual property goals, is found in their codebase. Employees must know where they can get answers regarding the use of a particular license, what to do if something appears to be unlicensed, what to do if licensing terms are complex, whether they are allowed to contribute to outside open source projects and so on. Knowing where these documents are stored can be part of the training program. Really well written, thoughtful documentation is only helpful if it is easily accessible. For some no cost policy and procedures suggestions, I recommend visiting the Blue Oak Council website. [9]

Producing a Bill of Materials for your products demonstrates to your own staff and your customers that you know what is in your code and that your company is serious about diligence in intellectual property and licensing matters. You cannot fix a problem if you do not know that it exists. For small codebases, this may be a manual effort. Larger codebases will require some type of discovery tool. Naturally, the larger the codebase, the greater the effort will be required to produce a bill of materials. A variety of tools exist to help with this problem, and they can be expensive, complicated enterprise tools, or they themselves may be open source. Smaller companies will naturally gravitate toward simpler, less expensive solutions or they may outsource the effort. Outsourcing tends not to scale up in terms of cost, so larger companies are more likely to have in-house staff. There are standards for Bill of Materials around open source, with SPDX being a key example with growing adoption for both manual and automated review processes[10].

Open source audits are not only crucial for M&A activity but should be incorporated into the software lifecycle. It can be understood that for smaller companies an M&A may be the only time an audit occurs, and it should equally be understood that it pays to be wary as such deals have failed due to a lack of diligence. Larger companies will want to keep continual tabs on their licensing exposure. Some companies are the proverbial Red Shirts of Star Trek fame. That is to say, they have a target on their back and parties aiming at that target may be hackers, customers, or even their competition. Although FOSS lawsuits are not very common, there have been occasions where licensing has been "weaponized" as in the case of CoKinetic Systems Corp. vs Panasonic Avionics.[11] Audits can be performed in-house, automated in the pipeline, or outsourced. There are an increasing number of tools, some free and some quite expensive, that can be employed for this effort. It is important to study the available tools and choose the appropriate ones given the time and money will be invested in implementing them. One of the best investments a company can make is choosing the right personnel to implement such programs, often those with a technical and legal background, who may have contributed to open source projects or be active in open source advocacy organizations.

7 https://training.linuxfoundation.org/training/beginner-guide-to-oss-development-lfd102/
8 https://www.youtube.com/watch?v=gF4b1TA5Q5w&list=PLAVikl6VpxPeBtplWOnfzNmiUz529AYAy
9 https://blueoakcouncil.org/
10 https://spdx.org
11 https://resources.whitesourcesoftware.com/blog-whitesource/the-100-million-case-for-open-source-license-compliance

Conclusion

Although open source compliance can be tricky and require resources it is not inherently necessary to employ dedicated personnel and enterprise solutions to accomplish a reasonable degree of best practices. This is particularly in situations where any resource allocation will compete for the resources required to actually create a product. This situation will often be the case with smaller companies and that is understandable that choices need to be made. However, compliance training courses, policy examples, and extensive information about effective processes can be found online. The first time a customer requests a Bill of Materials may be the catalyst for building such a program but it is hardly the sole reason for having one or regarding such activity as time limited. As a company grows, the compliance function should also grow to ensure ongoing effectiveness in the use of third party copyright for protection, for effectiveness and for solid positioning in both product and M&A activities. Over time and as experience grows the compliance function tends to move from legal review to operations and engineering. This saves the lawyers for the more difficult or unusual cases and reflects the fact that mature companies are expected to have trained personnel and processes in place. FOSS compliance is no longer an option – it is a necessity – and today it is a necessity that can be accomplished by organizations of any size and at any stage of growth with a little effort.

Further Reading

https://techcrunch.com/2019/01/12/how-open-source-software-took-over-the-world/

https://corporate.ford.com/legal/ford-open-source.html

https://www.zdnet.com/article/cern-leaves-microsoft-programs-behind-for-open-source-software/

https://arstechnica.com/information-technology/2017/09/massive-equifax-breach-caused-by-failure-to-patch-two-month-old-bug/

About the author

Robert Marion *is an Open Source Analyst and software engineer who has focused on open source compliance for over ten years and who has trained numerous corporations in multiple countries in open source compliance and compliance enterprise tools. Mr. Marion still writes software. When he's not working with code he enjoys playing guitar and touring the country on his motorcycle.*

Breathe In, Breathe Out: How open hardware licensing can help save the world

Jiri Svorc,[a] *Andrew Katz,*[b]

(a) Technology & IP Paralegal at Moorcrofts LLP, UK;
(b) Partner and head of Technology Law, Moorcrofts LLP, UK and Visiting Researcher, University of Skövde, Sweden

DOI: *10.5033/jolts.v10i1.139*

Abstract
As with any other open source field, there are countless far-reaching advantages in open hardware licensing, as opposed to its proprietary counterpart. This paper takes the example of a low-cost portable mechanical ventilator design and considers the effect of the application of the three different variants of the newly-released CERN Open Hardware Licence Version 2. This paper considers the importance of licensing, and demonstrates how open hardware licensing can facilitate efficient further development of a project, improve its safety and reliability, and encourage collaboration. Most importantly, open hardware licensing allows anyone to freely use, study, modify and distribute improvements to project design, and make, sell or otherwise distribute products made to that design, making it a cost-effective means of developing and deploying the device throughout the world, from the most developed to the most vulnerable territories. Finally, this paper argues that open hardware licensing also encourages economic activity whilst it protects third-party intellectual property rights.

Keywords
Open source; open hardware; licensing; CERN; ventilator; medical equipment; accessibility; CERN-OHL

Addressing a societal challenge

In 2010, a group of students from Massachusetts Institute of Technology and Boston University designed and prototyped a low-cost portable mechanical ventilator[1] that would help treat respiratory diseases, such as asthma or chronic obstructive pulmonary disease, in less developed countries. Although ventilators for artificial respiration have become commonplace in hospitals across many developed countries, they are provided at a cost of up to $30,000, with an equally high level of technological complexity. In their project, the students therefore set their sights on maintaining the

1 Abdul Mohsen Al Husseini, Heon Ju Lee, Justin Negrete, Stephen Powelson, Amelia Servi, Alexander Slocum, Jussi Saukkonen, *Design and Prototyping of a Low-cost Portable Mechanical Ventilator*, Proceedings of the 2010 Design of Medical Devices Conference (2010)

medical function of the ventilator, whilst reducing its price and making it easier to build. As a result, they developed a prototype whose bulk-manufacturing price was estimated to less than $200. Had the project been appropriately licensed and the ventilator began to be manufactured, it could have surpassed a great deal of similar projects[2] and helped doctors around the world saving lives of patients suffering from respiratory diseases, including from the recent outbreak of coronavirus disease COVID-19.

Although the students had planned to carry out further testing of their prototype and develop the idea so that it could be licensed for manufacturing,[3] they have only recently announced that they will make their material publicly available to help to find the solution to the global lack of ventilators in the COVID-19 pandemic.[4] Their announcement came only days after the initial submission of this paper. As more details about the project are being published day by dayand given the rapid pace of developments in the medical field these days, we have had to base our analysis on a number of assumptions made before the full disclosure of the project's details.

Given the project’s praiseworthy aim to increase availability and affordability of medical equipment and its initial lack of appropriate licensing, we have analysed the project to discuss the most appropriate licensing strategy for its fast, effective and large-scale deployment. Also, as the students had published the project report in an academic journal, we assumed that they intended to make the prototype freely and publicly available, as opposed to keeping it a secret and proprietary. We have seen this as an opportunity to consider the project in light of available open source hardware licences, namely the recently published version 2 of CERN-OHL.[5]

Do you need a licence?

Attaching a licence to a project that is intended to be made publicly available, so that it can be freely studied, used or possibly improved by anyone, may sound counter-intuitive but, in fact, it is necessary.

The rules of copyright law[6] automatically apply to a new work without registration and use by third parties is not possible without a licence, Ifanyone wants to copy or modify the project documentation they would first need to obtain (in the absence of any copyright law exception) the students’ permission to do so (assuming that they, and not their institution, were the copyright holder) . This may prove challenging in practice and would be given through the grant of a licence from the students.

. In some jurisdictions, including England and Wales, copyright laws could also prevent anyone without such permission from *creating a physical design* based on the project documentation.[7] Also, whilst the making and use of the ventilator based on the project may be allowed *for personal or non-commercial purposes* in some jurisdictions,[8] it may not be so in others.

Attaching a licence would therefore make clear to other designers and manufacturers if and to what extent they may study, improve, use and distribute the design, and manufacture and distribute

2 See https://www.instructables.com/id/The-Pandemic-Ventilator/ and https://medium.com/@RobertLeeRead/the-state-of-open-source-ventilator-projects-as-of-march-21st-1f36bfb608b4, both last accessed on 24 March 2020
3 See https://phys.org/news/2010-07-students-low-cost-portable-ventilator.html, last accessed on 24 March 2020
4 See https://e-vent.mit.edu/, last accessed on 31 March 2020
5 See https://ohwr.org/project/cernohl/wikis/Documents/CERN-OHL-version-2, last accessed on 30 March 2020
6 Other unregistered intellectual property rights such as unregistered design right, or database right, may also apply to aspects of the design materials, in various jurisdictions.
7 S. 213 and 226 of Copyright, Designs and Patents Act 1988
8 For example, see fair use provision of s. 107 of the United States Code. Likewise, Article 30(2) of Copyright Act of the Czech Republic stipulates that “copyright shall […] not be infringed by anybody who for his own personal use makes a fixation, reproduction or imitation of a work.”

products made to it.

What licence should you use?

Considering the mechanical features of the ventilator project and its aim to improve access to medical equipment, an open hardware licence (OHL), such as CERN OHL, appears most appropriate.[9] As an open source licence, an OHL would permit anyone to use the design materials from the project to make a ventilator themselves, either according to the design, or with any changes they decide to make to it.

This would facilitate rapid adoption of the equipment and give other professionals an opportunity to review it. They could identify and correct any imperfections or create enhancements, making it safer and more efficient to use as a result.

Most importantly, attaching an OHL to the project presents a cost-effective way of making the ventilator affordable in developing countries where the cost of currently available ventilators presents one of the most significant hurdles in their use. Needless to say, the far-reaching societal benefits of open source have been acknowledged by many[10] and open source licensing has consistently been recommended.[11]

Why CERN-OHL?

The European Council for Nuclear Research (CERN) has recently published the second version of its OHL licence, CERN-OHL. As one of the most respected and widely used open hardware licences, and being associated particularly with electronic devices, itis an appropriate choice for consideration. CERN-OHL offers three variants to choose from: a strongly reciprocal variant (CERN-OHL-S),[12] a weakly reciprocal variant (CERN-OHL-W),[13] as well as a permissive one (CERN-OHL-P).[14]

CERN-OHL-S

As a strongly reciprocal licence, CERN-OHL-S requires that any derivative design based on an original design licensed under it, is also licensed under CERN-OHL-S should it (or a product made to it) be distributed, like copyleft. It also requires that the licensee makes available with their design all design documentation of the derivative design, including the necessary installation and interfacing information.[15]

The licence recognises that designs of many items, from mechanical devices to electronic devices, often consist of generally and readily available components (“Available Components”).[16] Where this

9 See https://ohwr.org/project/cernohl/wikis/Documents/CERN-OHL-version-2, last accessed on 24 March 2020
10 See, for example, https://openuk.uk/, or https://www.gov.uk/guidance/be-open-and-use-open-source, both last accessed on 25 March 2020
11 The European Commission has recently opined that open source hardware could constitute a cornerstone of the future of Internet of Things (IoT) and the future of computing. See https://ec.europa.eu/digital-single-market/en/news/workshop-about-future-open-source-software-and-open-source-hardware, last accessed on 25 March 2020
12 See https://ohwr.org/project/cernohl/wikis/uploads/ee7922912e58f8676e1d7ff841b391cb/cern_ohl_s_v2.pdf, last accessed on 24 March 2020
13 See https://ohwr.org/project/cernohl/wikis/uploads/b94a1a92b29984226c56a0dd4dca0d39/cern_ohl_w_v2.pdf, last accessed on 24 March 2020
14 See https://ohwr.org/project/cernohl/wikis/uploads/055bd8b281d0805a3a38188838b370e1/cern_ohl_p_v2.pdf, last accessed on 24 March 2020
15 CERN-OHL-S, sections 1.3 and 1.8
16 Ibid, section 1.7

is the case, CERN-OHL-S does not (for physical components) require the licensee to provide exhaustive details of such components; it will suffice to provide enough detail so that they can be sourced and used to make the product or made themselves.[17] Of course, where a component is included that is not generally available, detailed information must still be provided.

Under CERN-OHL-S, the definition of Available Componentexempts the provision of interfacing and sourcing information and applies only to physical components. Therefore, if an original design is licensed under CERN-OHL-S then the licensor of a derivative design need not provide the full design documentation for any physical components available under a compatible licence, generally available physical components, or digital components available under a compatible licence.[18] Under CERN-OHL-S, readily available components which exist only in digital form, such as Hardware Description Language (HDL) cores, do not qualify as Available Components, so their Complete Source would have to be provided.[19]

Aside from being made of generally available physical components, such as a conventional bag-valve mask, cam arm, battery, motor and various tubes, the ventilator project also consists of an off-the-shelf Arduino Duemilanove microcontroller board[20] to control the functioning of the device.

The microcontroller on the Duemilanove (an ATmega 168 or ATMega 328) runs a simple piece of code: a logical loop where it responds to triggers ('*yes*' or '*no*') to prompt action and deliver intermittent breaths to the patient. While the Arduino Duemilanove itself qualifies as an Available Component of the overall design, how is our analysis affected when we take into consideration the code which runs in its microcontroller? Nothing changes. Code executed by a processor is not a component of that processor, the same way that the orange juice we use to fill a bottle is not a component of the bottle. The licensing regimes of the code and the hardware on which it runs are thus decoupled. Because the code is not a component of the hardware design, it is no necessary consider if it qualifies as an Available Component.

This leaves potential licensees with an important question: how can we make sure that code and hardware travel together, i.e that users always get a working ventilator? The answer is that licensing the hardware under CERN-OHL is not enough. There are two options. Either the licensor can also license the code explicitly under CERN-OHL (as a separate unit, or as a combined microcontroller+code unit), or (perhaps because the licensor is unable to make the code available under CERN-OHL, potentially because of licensing compatibility problems), they can apply another appropriate open source licence to the code.

If that code is a copyleft licence like GPL or LGPL, then on redistribution, the recipient is required to be given access to the source code under the same licence (it would, of course, be helpful if that source code were provided in, or through, a link available in the Source Location). In addition, the licensor would be well advised to give recipients additional comfort by seeking third party certification affirming that the project qualifies as open source hardware by complying with the OSHWA certification criteria, which stipulate that the hardware part of a project should be licensed under an Open Hardware licence and the software part should be licensed under a Free and Open Source Software licence.[21]

17 CERN, CERN OHL version 2 An Introduction and Explanation, available at https://ohwr.org/project/cernohl/wikis/uploads/0be6f561d2b4a686c5765c74be32daf9/CERN_OHL_rationale.pdf, last accessed on 24 March 2020
18 CERN-OHL-S, section 1.7
19 There is also an exception for components which are part of the normal distribution of a tool used to design or Make the Product. This acknowledges that many toolchains in the world of hardware are proprietary, and that they are likely to include items such as primitives, themselves proprietary, which will unavoidably end up in the design. This exception is mainly aimed at chip design, and this paper does not consider if further.
20 See https://www.arduino.cc/en/Main/arduinoBoardDuemilanove, last accessed on 24 March 2020
21 https://certification.oshwa.org/

Shouldn't we expect that software running on a board licensed under CERN-OHL-S be released under the CERN-OHL-S or another open source licence? No: and the reason for this is pragmatism. To explain, we provide an example.

It was recently revealed that many Intel x86 processors and their chipsets contain a microcontroller which runs a version of Andrew Tanenbaum's Minix operating system.[22] The Intel x86 processor chips themselves also run "microcode" which can be regarded as fundamental software which helps to execute the machine code instructions which the chips are designed to run. In each case, this is proprietary code. If there were a provision in CERN-OHL-S that *all* software and firmware running within the system must be open source, then this would prevent the use of almost any processor which uses microcode, including Intel x86 processors, and no doubt many other chips which also incorporate a small software stack. The reality is that these components are available to anyone, and users are provided with plenty of interfacing materials, so it would not be very useful to produce a hardware licence preventing a user from publishing designs using one of the most successful series of processors of all time. It would create even more serious problems, had a design for an Intel-based motherboard been made available under the CERN-OHL for several years, and become successful, before the discovery of the Minix stack in the Intel chipset had suddenly made the design retrospectively un-licensable.[23]

We already know that the Arduino Duemilanove qualifies as an Available Component under CERN-OHL-S, because anybody can buy one. For the sake of argument, let us see if it would qualify as an Available Component under another heading. Because the Arduino Duemilanove is itself open hardware, since the designs are freely available on the Arduino website, under a Creative Commons Attribution-ShareAlike (CC-BY-SA) licence, it might be considered that the Duemilanove is also available as Complete Source under a Compatible Licence. This is one of the ways a component can qualify as an Available Component under CERN-OHL-S (section 1.7(a)).

Unfortunately, this isn't the case, as CC-BY-SA is not compatible with CERN-OHL-S (or -W). Why? Because the CC-BY-SA requires that any changes to the design must, when distributed be released under CC-BY-SA (or a compatible licence[24]), and CERN-OHL-S requires that any changes to the design are, when distributed, released under CERN-OHL-S. Both of these requirements cannot be satisfied simultaneously.

Should Arduinos become distributed under CERN-OHL-S or -W (and we would ask that the rights holders of Arduino designs give serious consideration to dual-licensing them to enable this to happen), then the ventilator-custom Duemilanove would be capable of being regarded as an Available Component under section 1.7(a) ("licensed to You as Complete Source under a Compatible Licence").

CERN-OHL-W

CERN-OHL-W is similar to its strongly reciprocal counterpart, CERN-OHL-S, also requires that the licensee makes a great deal of information related to the derivative design available. However, it differs in its approach to virtual (including digital and software) components. While CERN-OHL-S

22 Steven J. Vaughan-Nichols, *MINIX: Intel's hidden in-chip operating system*, available at https://www.zdnet.com/article/minix-intels-hidden-in-chip-operating-system/, last accessed on 30 March 2020

23 A super-strong variant of the CERN-OHL which required that every piece of software and firmware within a design, including parts which were introduced by the tools, was suggested by people commenting on the licences during the drafting process, but it didn't seem that there were sufficient use-cases for this to be worthwhile. Maybe as hardware becomes more open in the future, this will be an option for a future licence.

24 CC-BY-SA 4.0 does contain a mechanism for allowing the out-licensing under a different compatible licence. The licences are selected through a process administered by Creative Commons. For example, it is possible to take a design licensed under CC-BY-SA and relicense it under GPLv3 (but not the other way around). It may be the case that CERN considers making an application to Creative Commons for one of both of the CERN-OHL reciprocal variants to be designated as compatible licences of CC-BY-SA 4.0.

only releases generally available *physical* components from strict information requirements, CERN-OHL-W extends this exemption to *any* component, including virtual ones.[25] As a result, where the derivative design incorporates a piece of code that is widely used and generally available (including under an open source software licence, for example), the licensee must acknowledge incorporation of such component in the project documentation, but need not include details about the making, testing, installation and interfacing of that code. This difference is most relevant in cases where the Open Hardware design in question is for an Application-Specific Integrated Circuit (ASIC) or a Field-Programmable Gate Array (FPGA). Since this article is about ventilators and similar hardware, we will not delve more into this side of things.

Another important difference between the -S and the -W variants is that a -W design can be merged into a larger design through a defined interface and the licensee doing this would not be expected to release the design details of the larger design. This is in contrast with the -S variant, whose strongly-reciprocal effect would result in an obligation to release the whole resulting merged design under CERN-OHL-S.

CERN-OHL-P

Finally, as expected, the "permissive" variant, CERN-OHL-P, permits the use, modification and redistribution of the design in any proprietary design. This may be particularly attractive to some businesses, as it allows them to develop the design and make products to it without having to release the design documentation. It's important to realise that designs licensed under any variant of CERN-OHL can be produced in a commercial context: none of the licences prevent commercialisation of the product, but the reciprocal variants do require the design documentation to be made available (and potentially used by competitors).

Allowing commercial use is justified

There are numerous alternative licences that may be considered for licensing the ventilator project, such as Creative Commons Non-Commercial licences (CC-NC). By explicitly excluding re-use scenarios leading to monetary compensation or other commercial advantage, these licences appear to serve public interest. However, this is not necessarily the case.

Licensing a project under a CC-NC licence only imposes non-commercial use on the subsequent use of the project by third parties, not its commercial exploitation by the original rights holder (assuming the original rights holder holds *all* the rights, and has not acquired some of the rights through the involvement of a community of contributors who have themselves contributed back to the project under a CC-NC licence).

As a result, the original licensor may effectively become the exclusive commercial user of the project and protect its commercial interests.. This may also negatively impact the ability of a community to coalesce around the project. It also creates the added complexity thatany improvements which are made to the design and which are re-submitted to the project under an NC licence, cannot be used by the original licensor (or anyone else) on a commercial basis.In addition, the terminology of CC-NC is subject to some dispute as to what constitutes a commercial advantage; this can range from profits to reputation, short-term to long-term.

In the current crisis,[26] it is important to maximise the production of high-quality ventilator designs as quickly as possible. To add a hurdle to commercial organisations – the very organisations with tooling and expertise to produce the ventilators –does not make sense, since there is no shortage of

25 CERN-OHL-W, section 1.7
26 Coronavirus disease (COVID-19) pandemic, see https://www.who.int/emergencies/diseases/novel-coronavirus-2019, last accessed on 30 March 2020

demand.

It's also important to note that in the world of software, the raw materials – zeroes and ones – are free of charge and in infinite supply. Hardware, by definition, requires atoms which (almost always) need to be purchased, and therefore, it is almost impossible for an open hardware project not to involve the injection of commerce at some point.

For these reasons it makes sense for a licence which permits commercial use to be relied on, but also one which encourages the sharing of designs and their improvements (including production engineering improvements, and market-specific improvements), and allows suitably equipped makers to collaborate with minimal friction to provide a suite of the best designs available for different markets, different applications, and different locations. Either the CERN-OHL-S or CERN-OHL-W would be ideal for this purpose.

Patents

The Creative Commons licences expressly exclude patent licensing, which means that a participant in the ventilator project which holds patents could simultaneously license the copyright in their designs in an open way, and at the same time withhold any patents necessary to produce and use those designs. This would seem unfair and unreasonable, and the CERN-OHL licence suite is intended to address this issue. It does so in the same way as many more modern open source software licences, by both providing an explicit patent licence covering contributions made to a project, and also with a patent retaliation clause which removes rights granted to a licensee should they start attacking a licensor for patent infringement in relation to the design.

Other features of the CERN-OHL-S and -W

A particularly attractive feature of the reciprocal versions of CERN-OHL is the requirement that a licensor may apply to ensure that details of a Source Location are provided on any Product made to the design, whether on the design itself (for example, a short-form URL placed onto an object as part of the 3D printing process, or silk-screened onto a circuit board), or on its packaging or documentation. This requirement may be particularly powerful if the Source Location details are placed onto the ventilator itself. If anyone can easily track down the design documentation, they can also use it to troubleshoot and fix any problems, even if they do not intend to replicate the entire device.

Conclusion

We recommend that any projects releasing open hardware designs for ventilators should give careful consideration to licensing them under CERN-OHL-S or CERN-OHL-W (either v2 or any later version). We also suggest that they consider licensing any necessary software or firmware under the same CERN-OHL licence, either separately, or as part of the whole design. If that is not possible, then we recommend consideration is given to an appropriate copyleft licence such as a version of the GPL or LGPL.

We contend that this enables commercial entities to collaborate easily to create the physical product, while at the same time allowing appropriate protection for patents, third parties and users. It also potentially means that, should the same reciprocal licence be employed, it becomes very easy to mix components between different designs where necessary, and also to potentially locate the relevant information necessary to maintain, fix and operate the devices in the field.

About the authors

Jiri Svorc *works with Moorcrofts LLP as a Technology & IP Paralegal, having previously worked with law firms in the Czech Republic, the Netherlands and Luxembourg as well as with the European Commission in Belgium. In various roles, he participated in advising businesses as well as public sector organisations on technology, intellectual property and commercial law matters as well as personal data protection and cyber security. Most notably, he has participated in cross-border technology transfers, and advised the largest global technology companies as well as developers of disruptive technologies. Jiri has a keen interest in open source software (he has written his postgraduate dissertation on using open source code in proprietary software), key specialisms of the Technology Team at Moorcrofts and rapidly expanding fields in the industry and has been working with Andrew Katz in this area. Jiri holds a postgraduate qualification from Queen Mary, University of London, has completed his legal practice exams, and plans to qualify as a solicitor.*

Andrew Katz *is the Partner and head of Technology Law at Moorcrofts LLP, UK, and head of the Technology Department. He has been practising technology law for over 20 years, having previously been a programmer and accredited NeXT developer. He has a particular interest in free and open source software, open data and open hardware. He studied Law at Cambridge University, qualified as a barrister, and subsequently re-qualified (and practises) as a solicitor in England and Wales. He is also a solicitor (non-practising) in Ireland. He lectures and works extensively worldwide. He was on the core drafting team of the CERN-OHL 2.0, drafted the Solderpad Open Hardware licence, and advises the European Commission on Open Hardware policy. His work has been published by Oxford University Press, Edinburgh University Press and others. He is deputy chair of the IP and Open Source advisory group of the United Nations Technology Innovation Labs, a visiting researcher at the University of Skövde, Sweden, where he has co-authored several papers, one of which has been used as the basis for Swedish government procurement policy. His most recent publication (2020) analyses the applicability of Roman Law, to the internet, with specific focus on autonomous intelligent agents.*

The authors acknowledge and thank Javier Serrano of CERN for his helpful commentary and review of this paper. Any errors or omissions, however, remain those of the authors.

Linux Foundation is Eating the World

C. Bradford Biddle [a]

(a) Faculty Fellow, Center for Law, Science and Innovation, Arizona State University, and Principal, Biddle Law PC, Portland, OR USA

DOI: 105033/jolts.v11i1.137

Abstract
Marc Andreesen's 2011 article Software is Eating the World suggested that developments in the information and communications technology (ICT) industry are now transforming industries far beyond ICT. By facilitating interoperability, private sector-led technology consortia contributed to these developments, and they will continue to play a critical role as interoperability requirements grow in complexity. Consortia themselves have evolved over time, and continue to change. While historically many consortia focused on hardware interoperability, open source software is increasingly part of how interoperability occurs, and today's consortia reflect this. The extraordinary growth and rapidly expanding roles of the software-centric Linux Foundation is striking evidence of this new reality. This story holds important lessons for European stakeholders. Within this changed technology standardization landscape there are opportunities for European leadership.

Keywords
Law; information technology; Free and Open Source Software; standards; consortia; interoperability

I. Introduction

An explanation of the title of this article can serve to explain its goals. Linux Foundation is a non-profit organization based in the USA that hosts development of the open source operating system software called Linux. Less obviously, Linux Foundation also hosts at least 155 other collaborative software and specification development projects, and is growing at an extraordinary pace. The title also references a well-known 2011 article by venture capitalist Marc Andreessen called Why Software is Eating the World. In this article Andreessen argued that "six decades into the computer revolution, four decades since the invention of the microprocessor, and two decades into the rise of the modern Internet, all of the technology required to transform industries through software finally works and can be widely delivered at global scale." (Andreessen, 2011). He describes a software revolution transforming a broad range of industries: entertainment, retailing, manufacturing, health care, education – even industries like oil and gas and national defense.

This paper argues that that private sector-led collaborative technology development organizations called "consortia" have been a fundamental part of advancing this revolution. It further argues that they are now also being transformed by it, and the remarkable growth of Linux Foundation is evidence of this transformation. Ultimately this is not a paper about the Linux Foundation, however. Rather, the paper explores more generally how the technology standardization process is changing, and surveys what these changes may mean in particular for various European stakeholders.

Consortia such as the USB Implementers Forum, the Wi-Fi Alliance and the Bluetooth SIG, and countless other similar organizations have long played a critical role in developing interoperability standards in the information and communications technology (ICT) industry – arguably a more important role than formal international standards development organizations like the International Telecommunications Union (ITU) or the International Organization for Standardization (known as ISO). Over the past several decades the ICT industry developed and honed a model for the formation and operation of collaborative groups that enabled ICT product interoperability in a diverse array of technology areas. Typically the core deliverables of these organizations have been technical specifications – textual documents that describe how to build interoperable systems – which in many cases serve as standards in particular industry segments.

This specification-focused model is increasingly challenged by open source software projects, such as those hosted by Linux Foundation, that facilitate interoperability through shared software code rather than shared technical specifications. This challenge has forced traditional consortia to adapt, with software now playing a more important role in many organizations. At the same time, both traditional consortia and open source software projects seem to be recognizing some limitations of software as a path to sustained industry interoperability (in part due to the risk of software projects "forking" into incompatible paths), and appear to be seeking some synthesis of the specification-oriented and the software-oriented models.

We are in the midst of an era of significant change around how technology consortia organize themselves and the types of deliverables they produce. And, because consortia play a critical role in the global ICT standardization process, and because the ICT standardization process has implications far beyond just the ICT industry, this change is consequential.

Historically technology consortia have been largely a United States-based phenomenon. Of the many hundreds – perhaps thousands – of private sector-led technology consortia that have been formed over the past decades, only a small percentage have been based in Europe. Europe has certainly played a leading role in global standardization – in addition to being the home of formal standards organizations like ITU and ISO, European-based organizations like 3GPP, ETSI, ECMA and others are major forces in ICT standardization – but European consortia in the style of USB, Bluetooth, Wi-Fi and the like are the exception, not the rule.

The European Commission recently issued a call for a study "to reinforce the EU's competitiveness in digital technologies," based on an "observation that European industry needs to come to agreements on functions and interfaces for those platforms, reference architectures and interaction protocols that have the potential to create markets and market opportunities leading to ecosystems and standards" (European Commission, 2018). Fundamentally, this is what specification-oriented technology consortia have done for decades, and what collaborative open source software projects like those hosted by Linux Foundation are increasingly doing. The EC study proposal suggests that European policymakers are rethinking the role that European industry and European institutions play in this process. Given the transformational effect technology is having across many different kinds of industries, and given the changing environment of private sector-driven collaborative interoperability efforts, this reevaluation is both important and timely.

The paper proceeds in three main sections. The first part explains the critical role played by technology consortia. The second part identifies how consortia are changing. The final section offers some observations for European stakeholders to consider in light of this changing environment.

II. The Importance of Consortia

The technology consortia that are the focus of this paper occupy a space between, on the one side, unilateral actions of a single company, and, on the other side, formal standardization efforts. This paper adopts the taxonomy of "single company" efforts, "consortia," and "formal standards development organizations" set forth in Biddle (2017); the emphasis here is on consortia, under that framework. Accordingly, this paper focuses on structured, private sector-led collaborative efforts that produce technical specifications, software code, reference designs, or other similar deliverables that enable interoperability between third party products or services, or that otherwise support such efforts by providing compliance testing services, marketing support, or similar support services. Technology consortia are the organizations that are formed – with varying structures and degrees of formality – to advance these efforts. Formal standards development organizations – i.e., those organizations vested with some direct or indirect governmental authority to create "standards," including all ANSI-accredited standards developers in the U.S. – are definitionally not consortia. In practice, the boundaries between these various kinds of activities can be blurry, however: some large consortia look much like formal SDOs, and some smaller or more specialized SDOs look much like private sector-driven consortia.

Examples of well-known technology consortia include:

- *USB Promoters Group and the USB Implementers Forum.* The USB Promoters Group is a small group of "promoter" companies that establish contractual relationships with a larger group of "contributor" companies and develop the primary Universal Serial Bus specifications. The promoter companies then enlist the tax exempt non-profit corporation USB-IF to distribute the specifications to "adopters," to manage a compliance testing process and associated certification and logo-licensing program, to market the USB specifications to potential adopters, and to perform various other related services. USB is "the most successful computer interface ever," according to NV (2014).

- *Bluetooth SIG.* The Bluetooth SIG was organized as a non-profit corporation under applicable U.S. state law, although its tax exempt status was rejected by the U.S. federal government. The entity is led primarily by corporate directors appointed by a handful of "Promoter Member" companies, comprised largely of the companies that founded the organization. The Bluetooth SIG coordinates development of the Bluetooth specifications, and runs an associated compliance testing process. The organization reports that there are over 30,000 implementers of the specifications.

- *Wi-Fi Alliance.* The Wi-Fi Alliance is a non-profit corporate entity that formed initially to provide compliance testing and promotional support for certain wireless local area network standards produced by the large formal standards body IEEE. The entity is led by directors appointed by a group of sixteen "sponsor members." Wi-Fi Alliance now produces its own specifications, in addition to continuing to provide compliance testing and marketing for this family of IEEE standards.

Beyond these large, relatively well-known organizations, countless other industry-led collaborative efforts engage in similar activities, with both large and small impacts. The PCI-SIG's PCIe specification is used in virtually every high end computing device. The HDMI Forum defines the ubiquitous HDMI video connector. MIPI Alliance produces interface specifications for mobile device hardware that are in embodied in literally every modern smartphone on the planet. The OPC Foundation's OPC UA specification and associated software code is ubiquitous in the industrial automation industry and is shaping the factories of the future in profound ways. Khronos Group provides the foundational specifications and code that enable cross-platform virtual and augmented reality. OpenStack Foundation's open source code has transformed the data center. Groups fill narrower niches too: the QSFP-DD MSA Group, for example, is a contractual arrangement between various stakeholder companies that was formed to create a specification for a new optical hardware connector used in telecom routing devices. While no one knows the exact number of these kinds of groups, it is fair to state that they number in many hundreds, and perhaps thousands.[1]

Consortia have played a critical role for the information and communications technology industry. Perhaps more than any other industry, ICT faces deep and complex needs for interoperability between third party products and services. ICT products are built from tens of thousands of components sourced from an equally large number of vendors. These products leverage complex firmware, operating system and user-level software stacks, and they communicate over a variety of local and wide-area wired and wireless networks that ultimately span the globe. They are employed in a near-infinite variety of use cases. The associated requirements of coordination between a vast array of different actors are staggering. Consortia are a principal tool used by the ICT industry to create and manage interoperability in this extraordinarily complex context.

In addition to facilitating sophisticated supply chains and product use cases, interoperability in the ICT industry also results in positive network externalities that are likely impossible to measure in the aggregate but that are undoubtedly immense in effect. One recent study (Huawei 2018) suggested that "intelligent connectivity," which it described as "more integrated connections between all things, machines, and people in industrial settings," would produce US$23 trillion in new economic potential by 2025. Consortia are a key forum in which these kinds of connections are defined and developed.

Consortia are arguably more important for ICT interoperability than formal standards organizations. Formal standardization is a long, slow process that is most effective when marketplace consensus has already been established. Consortia are sometimes the battlefield where this kind of consensus is fought for and won. Groups rise and fall depending on the level of marketplace support they garner. The BluRay Disc Association competed with the DVD Forum's HD-DVD specification, until the market tipped to BluRay. The Wireless Power Consortium competes with AirFuel Alliance with different visions for wireless charging, while Airfuel swallowed the Power Matters Alliance. Leading industrial automation groups recently negotiated a compromise that blends their various visions. These examples demonstrate that consortia allow a type of market dynamism that is muted at the formal standards level. Frequently the technologies that are brought to the formal standardization process are those that have been developed and achieved marketplace acceptance via consortia.

Consortia also can complement formal standardization processes in other ways. Formal standards development organizations typically produce one deliverable: technical specifications intended as standards, embodied in descriptive textual documents. For example, the IEEE produced the 802.11 wireless communications standards. The Wi-Fi Alliance gave this technology a consumer-friendly name – Wi-Fi – and marketed it so that consumers understood the value of buying a product that had Wi-Fi functionality. Wi-Fi Alliance also did the critical work of developing a testing process that ensured that W-Fi products actually interoperated with each other in real-world implementations, and developed a sophisticated logo licensing model that enabled consumers and supply chain participants to accurately communicate that their products worked as advertised. The importance of the marketing and compliance testing activities, beyond the bare publication of the standard, cannot be overstated. Other consortia play this sort of complementary role to particular formal standards; for example HomeGrid supported the ITU's G.Hn standard in this manner.

Quantitative assessment by researchers evaluating the role of consortia in technology standardization has been relatively rare. Biddle et. al. (2010) identified 251 standards in a then-current laptop computer and found that a 44% of these were developed by consortia (along with 36% developed by

1 The website consortiuminfo.org lists over 1000 organizations, but some are formal standards development organizations rather than consortia. This author maintains a database that includes additional consortia that are not listed at consortiuminfo.org. Others likely exist that are included in neither database. This author's best guess is that with modest effort one could specifically identify about 1000 past and current ICT consortia.

formal standards setting organizations and 20% promulgated for industry adoption by single companies). Armstrong et. al. (2014), in a paper focused on estimating patent royalties, identified key standardized technologies in a smartphone. Drawing from the Armstrong et. al. (2014) work, and categorizing the standards developer as either a consortium or a formal standards development organization, the breakdown would be:

Consortia (11 organizations)	**Formal SDO (5 organizations)**
SD Card Association (memory)	3GPP (Wireless WAN, wireless protocols)
WiFi Alliance (802.11 compliance)	IEEE (802.11)
Bluetooth SIG (Bluetooth)	ITU-T (Wireless WAN, imaging)
NFC Forum (near-field wireless)	ISO/IEC JTC1 (audio, imaging, more)
MIPI Alliance (camera, display, more)	JEDEC (memory)
Open Mobile Alliance (MMS)	
Open Handset Alliance (Android OS)	
Internet Engineering Task Force (various internet protocol interfaces)	
World Wide Web Consortium (various web interfaces)	
UPNP Forum (local networking)	
Digital Living Network Alliance (content)	

Similarly, using the same criteria, one can categorize the thirty six organizations identified by Baron & Spulber (2018) in their sample of leading ICT standards setting organizations as follows:

Consortia (22 organizations)	**Formal SDO (14 organizations)**
Accelera, CEA, DMTF, DVB, HomePlug, HomePNA, IETF, IMTC, IrDA, MEF, OGC, OMA, Open Group, OSGi, PCCA (WTA), PCI-SIG, PICMG, SDR Forum, VESA, VITA, WIFI	3GPP, ANSI, ASTM, ATIS, BioAPI (ISO/IEC JTC1), CEN, ECMA, ETSI, IEEE, ISO, ITU, JEDEC, OASIS, TIA

This data provides some empirical support for the argument that consortia play a fundamental role in technology standardization, exceeding even the role played by the better-known formal standards organizations. Formal SDOs created only about a third of the standards in a 2010-era laptop, while consortia created nearly half. In connection with identified smartphone standards, consortia were named as standards developers twice as often as formal SDOs. About 60% of the standards setting organizations selected by scholars studying a collection of important ICT standards are consortia, and only about 40% are formal SDOs.

Qualitatively, the critical role played by technology consortia in facilitating interoperability for the ICT industry seems indisputable. Examples like USB and Bluetooth – along with hundreds of lesser-known organizations – show how consortia create important standards-based market ecosystems. As suggested above, consortia also play a unique pre-standardization role, enabling market participants to group themselves around potentially competing technologies, collaboratively developing technologies that may eventually become formal standards. And, as discussed using the example of the Wi-Fi Alliance, consortia sometimes complement formal standards by providing necessary compliance testing and marketing functions that formal standards bodies do not provide. Consortia are a fundamental part of the story of why, in Marc Andreessen's words, information technology "works and can be delivered at a global scale."

III. How Consortia are Changing

Consortia are changing in two important ways. First, new structural models, embodying new ways of forming and governing consortia, have emerged and have rapidly altered the consortia landscape. Second, the nature of what consortia produce is changing. Software increasingly plays a primary role. Linux Foundation is a leading example that illustrates both of these major changes.

A. Structural and Organizational Changes: Continuous Evolution

This section of the paper describes how the structural model for consortia has evolved over time to address a complex set of legal and practical concerns, such as managing liability risks for participants, mitigating the risks of antitrust or competition law claims, negotiating difficult intellectual property and governance/decision-making questions, reducing taxes, managing finances, and addressing a broad range of operational matters. This evolution is discussed in some depth, accompanied by a detailed analysis of several examples, in part to frame later discussion about how European stakeholders might address these same issues. A reader who is less interested in the minutia of how U.S.-based consortia have organized themselves could reasonably skip this section, taking away the general point that structural considerations are generally driven by these kinds of important underlying concerns.

ICT consortia date back to at least the 1980s. Cargill (1989) discussed the emergence of the Corporation for Open Systems (COS), the Manufacturing Automation Protocol (MAP), and other consortia (using the term "consortia" in essentially the same way as the term is being used in this paper). A 1989 magazine article described a "proliferation of computer industry standards groups, which makes rabbits look positively abstemious by comparison." (CBR Staff Writer, 1989). Mähönen (1999) described an environment similar to that described in this paper:

> *Especially in the field of telecommunications, standardization used to be the province of international organizations such as ITU (International Telecommunications Union), ISO and IEC (International Electrotechnical Committee). Now, the activities in telecommunications, information technology and multimedia are also addressed by a multitude of other players in the field. The standardization organizations can now be categorized into two main groups: formal (de jure) and informal consortia (de facto, grey or ad hoc groups). The formal standardization processes arc handled by traditional standards development organizations (ISO, ITU etc.), scientific or professional societies, trade associations or industrial standard organizations that can have a liaison with formal official bodies. Informal standards, in contrast, are produced by market forces (de facto) or by specific groups or consortia working independently.*

By 2000, a dominant structural model for consortia had solidly emerged. The USB promoters group, which had been formed and distributed its initial 1.0 specification in 1996 under a purely contractual arrangement between stakeholder companies, coordinated the formation of the USB Implementers Forum Inc. as a mutual benefit (i.e. trade association-style, as opposed to a public charity) non-profit corporation under applicable U.S. state law (Oregon). Bluetooth SIG and PCI SIG, both of which had begun initially under similar contractual-based models, followed suit and developed formal incorporated structures in that same year. USB-IF and PCI-SIG successfully established themselves as tax exempt non-profit corporations recognized by the U.S. federal tax authority, the Internal Revenue Service (IRS). (Interestingly, Bluetooth SIG fought and lost a battle with the IRS over

Bluetooth's tax exempt status.[2])

Many other organizations formed following this same basic template: incorporation as a mutual benefit non-profit corporation under applicable U.S. state law (with some slight variations of corporate form based on particular state law requirements), and then operation as a tax exempt entity under a provision targeted at "business leagues" and other trade association-style enterprises. This provision, Section 501(c)(6) of Title 26 of the U.S. Code, generally enabled the organizations to avoid paying federal income tax, and often to avoid most state and local taxes as well. Selecting from hundreds of examples, some organizations that follow this model include Avnu Alliance, the Broadband Forum, CCIX, the DASH Industry Forum, the Ethernet Alliance, FIDO Alliance, GENIVI Alliance, HDBaseT Alliance, ID Federation, JEDEC, Kantara Initiative, LoRa Alliance, MEMs Industry Group, NVM Express, Open19 Foundation, PICMG, Risc-V Foundation, SATA-IO, Thread Group, Universal Stylus Initiative, VESA, WiFi Alliance, XBRL and Zigbee (alas, we found no "Y" example of a 501(c)(6) org, defeating our attempted A-to-Z list – but we'll mention the Yocto Project as an example of a different model below).

These organizations are typically funded by membership dues, sometimes supplemented by revenue from other sources, such as a compliance testing program that requires the payment of fees. Often there are tiers of membership, and members at the higher tiers frequently have strong influence over organization governance. For example, in the LoRa Alliance only "sponsor members," who pay a US$50,000 membership fee, are eligible nominate and to vote for the organization's corporate directors.

Prior to the emergence of this incorporated model for consortia, many groups had formed under multi-party contractual arrangements. One model, popularized by USB prior to formation of the USB Implementers Forum, and followed by many others, was to identify "promoter" companies that led an organization, "contributor" companies that provided substantive inputs but lacked final decision-making power, and "adopter" companies that implemented the group's deliverables, with the companies executing either a Promoter Agreement, Contributor Agreement or Adopter Agreement as applicable. While still used occasionally, this model became disfavored for three primary reasons. First, under applicable U.S. law the participants in these groups faced the risk of "joint and several liability" for the actions of other participants – i.e., one deep-pocketed participant could be held liable for the actions of a third party group member. Second, the lack of a distinct legal entity created various practical problems: the groups could not open a bank account, or enter into contracts, or apply for and own trademarks or other intellectual property – all of which proved to be important activities for many organizations. Third, these direct agreements to cooperate, made between parties that often were otherwise competitors, raised questions about potential antitrust (competition law) liability.

The incorporated organization model addresses all of these problems. Under applicable law, absent extraordinary circumstances, members are insulated from liability for the activities of the corporate entity. Further, as an independent legal entity the incorporated body can manage funds and hold intellectual property, and enter into contracts. As an independent non-profit entity the organization also offers its members a stronger narrative around competition law questions, as well as the possibility of taking advantage of certain liability safe harbors that applicable U.S. law offered to more traditional standards organizations.[3]

The incorporated model has some drawbacks as well. Forming new organizations can be contentious

2 The case is nicely summarized in Bluetooth SIG, Inc. v. United States (2010). To this author's surprise, the case appears to have had little impact on the tax treatment of other U.S. technology consortia.

3 The National Cooperative Research and Production Act of 1993 provides certain antitrust liability protections to joint ventures and the Standards Development Organization Advancement Act of 2004 extended the provisions of the NCRPA to standards development organizations. The precise application of these statutes to consortia raises complex questions that won't be addressed here. As a practical matter, whether as a result of these statutes or otherwise, generally consortia seem to avoided significant antitrust scrutiny.

and time-consuming, as each new organization requires a new negotiation of governance details and intellectual property licensing arrangements. In the context of this type of organization, these negotiations can be a proxy for broader competitive dynamics in an industry segment. An influential party may wish to exclude a competitor from a leadership position, for example, or may want to ensure that the head of a technical committee is an ally. A decision between a royalty free (RF) or a fair, reasonable and non-discriminatory but potentially royalty-bearing (FRAND) intellectual property rule can fundamentally alter business models in a particular industry segment. Given these high stakes, among parties with potentially very different business interests, it is unsurprising that formation negotiations sometimes prove difficult; but still, the frequent contentiousness and delays frustrate the affected parties. Further, once formed, groups require operational support and compliance with various tax and legal requirements that can prove burdensome. These difficulties have left industry participants hungering for an easier-to-implement model.

The IEEE Industry Standards and Technology Organization (ISTO) was an early attempt at a more efficient model. Founded in 1999 as a 501(c)(6) non-profit corporation fully independent from the IEEE, ISTO focused primarily on solving the operational support and compliance issues faced by independent ICT organizations. ISTO offered groups a corporate umbrella under which they could organize largely autonomous projects, with operational support from ISTO staff. When forming, however, each project still faced difficult negotiations over project governance and intellectual property, as each project developed its own unique charter documents. The ISTO model also raised new questions around liability and legal autonomy. For example, a large ISTO project with significant financial resources might worry that a different ISTO project could create a legal liability for ISTO that could drain the first project's resources. A few ISTO projects ultimately formed separate corporate legal entities to address this risk – a step which undermined some of the ostensible simplification benefits offered by ISTO. Ultimately, however, ISTO appears to have established an important niche in the ICT industry: ISTO reports that it has supported over 50 groups in its 20 year history, and it currently lists 17 active groups on its website. Most of these appear to be simply projects of ISTO, rather than separately incorporated entities, but apart from this structural difference these projects look and act like other independent technology consortia.

The Joint Development Foundation (JDF) is a more recent example of an attempt to improve the process of organizing consortia. Founded in 2015, JDF is also a non-profit corporation with tax exempt status under Section 501(c)(6). JDF's goal was to provide groups what it called a "consortium in a box." Like ISTO, JDF provided sub-contracted operational support to groups (as and if desired by the groups), but JDF's focus was on simplifying the legal details associated with group formation, largely by providing a set of menu options of well-defined legal terms. JDF described its value as follows:

> *By using established Joint Development Foundation legal agreements, groups can establish projects quickly and with minimal legal expense. By operating under the Joint Development Foundation's legal umbrella, Projects can enjoy of the benefits of the Joint Development Foundation's existing legal agreements, choice of intellectual property policies, non-profit status, and corporate structure. This enables Projects to more easily establish themselves, collect funds, issue press releases in the Project's name, develop liaison relationships, and hold copyrights, all without negotiating custom agreements and new corporate organizations.*

JDF also developed an innovative legal structure for its groups, conceived by JDF founder David Rudin. It formed a subsidiary legal entity, called Joint Development Foundation Projects LLC, as a single member limited liability company (LLC) under the state law of the U.S. state of Delaware. In turn, JDFP LLC was empowered, under an applicable provision of Delaware law, to create "series LLCs" – essentially simple-to-form subsidiary entities of JDFP LLC. Each JDF project was assigned its own series LLC. For example, the large JDF project known publicly as Alliance for Online Media is technically the "Joint Development Foundation Projects LLC Alliance for Open

Media Series." The entity then filed to do business under the trade name "Alliance for Open Media." This model enabled each project to have its own distinct legal entity for purposes of contracting and as a wall insulating against liability: theoretically, liability created by one series entity cannot affect another series LLC, the parent LLC, or the ultimate parent corporation. For tax purposes, however, the LLCs are considered "disregarded entities" by the U.S. federal tax authority, because they each have a single legal member, the parent JDF corporation. Thus for tax purposes they all are treated as JDF – that is, as part of a tax exempt 501(c)(6) organization – and they generally are not obligated to pay federal taxes.

Between 2015 and 2018 JDF launched four public projects. In 2018 JDF and Linux Foundation announced a plan to "bring the Joint Development Foundation into the Linux Foundation family." Currently JDF and the JDF projects are identified as Linux Foundation projects on the LF website.

Linux Foundation itself was founded in 2000, with an initial focus specifically on the Linux operating system. By the early 2010's it had established a program it called "Linux Foundation Collaborative Projects," under which it hosted other projects. In August of 2013 it hosted nine such projects, including Tizen, the Xen Project, the Yocto Project, and others. Today Linux Foundation hosts at least 156 projects. Most Linux Foundation projects are different in two important ways from the vast majority of the consortia discussed in this paper thus far: (a) the LF projects primarily produce open source software code rather than technical specifications, and (b) the governance and intellectual property models follow open source community norms, which differ from the norms of traditional specification development groups. These issues will be discussed further below. For now, focusing on organizational structure, the key point is that for much of the history of Linux Foundation the Foundation followed the ISTO model with its projects: that is, most were simply projects of LF, without any formal separate legal identity – although a few were separately incorporated as 501(c)(6) non-profit corporations. However, beginning in 2017, many LF projects now reference the series LLC structure; for example, the FD.IO Project is "FD.IO Project a Series of LF Projects, LLC." Accordingly, it appears that, in addition to now directly incorporating JDF and its projects as of 2018, LF has also emulated JDF's structural model, presumably to achieve similar goals of liability insulation, ability of groups to independently hold funds and intellectual property, to contract directly with third parties, etc. Plus, like ISTO, LF offers its projects a sophisticated set of services, ranging from website design and hosting to event planning to finance, operations and human resources support, to compliance program develop and implementation, and more.

All of this discussion is intended not as a comprehensive explanation of how consortia are structured – in fact, exploration of some important and interesting variations, such as the approach followed by various content protection groups like Digital Content Protection LLC, the group that created the HDCP specification, are omitted here – but rather to illustrate the point that the models for how technology consortia are formed and structured have followed a complex evolutionary path, reflecting an array of legal, tax and operational factors, and that these models continue to change. Some fundamental structural innovations have appeared just in the past several years.

B. Changes in Deliverables: Software Becomes Increasingly Important

The discussion in this section of the paper is intended to illustrate that software has increasingly become a key tool for creating interoperability in the ICT industry. Further, the development methodologies, intellectual property models, and ultimately the culture of open source software appear increasingly important in the development of interoperability solutions.

Organizations like USB, PCI-SIG and other traditional consortia primarily produce technical specifications. These are textual documents that describe how to build interoperable products. Engineers read these documents, and build products accordingly. Consortia frequently additionally offer other supportive services, such as "plugfests" (informal forums where engineers could test their products with others), or more formal compliance testing services, or marketing services to promote the value of interoperable products, but fundamentally the core deliverable of many technology consortia are technical specifications.

Historically these specifications typically described implementations in hardware. Building hardware requires careful long-term planning, and once commitments to particular technical paths are made they are difficult or costly to reverse. Accordingly, organizations focused on creating technical specifications defining interoperability for hardware products typically emphasize specification quality and organizational consensus over development speed. Development methodologies generally typically follow disciplined systems engineering approaches.

Open source projects primarily produce software code. Examples of hugely successful open source software projects include the Linux operating system, the Apache web server, the Firefox web browser, the MySQL database and countless other widely-deployed components and applications. Open source software is, by definition, licensed under an open source license. Open source licenses grant licensees broad rights to re-use code. Most modern open source licenses include express royalty-free patent licenses applicable to contributed code, and many open source community members argue that such licenses are implied even when they are not explicit. Further, at the risk of severe over-simplification, open source projects generally adopt a governance/decision-making model that relies more on meritocracy than hierarchy, and that permits open participation in a project.

As "software eats the world," increasingly some interoperability problems that used to be solved in hardware are now solved in software. A leading example is the rise of software-defined networking (SDN) and network function virtualization (NFV) in telecommunications. Functions that historically were performed by hardware-based switches, controllers and data plan infrastructure now are performed by open source code produced by groups like Open Daylight, OpenSwitch, and FD.IO.

Traditionally consortia created technical specifications and then users took these specifications and developed their own implementations (e.g., built their own hardware devices). Software presents the opportunity for collaborators to simply develop a shared implementation, making that implementation available to all potential implementers as open source software code.

Software has also become more important as ICT and other industries focus on broader systems-level interoperability, which is another underlying component of the Marc Andreesen vision. As a prescient U.S. Department of Defense report stated the issue: "System interoperability is what makes heterogeneous systems of systems a reality. All of these systems are composed of hardware and software. Hardware is not easily changed. Furthermore, fielded hardware systems often cannot be wholly replaced. Therefore, as a practical matter, interoperability is more easily achieved through software…" (Hamilton & Murtagh, 2000). This shift towards software also enables new opportunities, as stated by Carney et. al. (2005): "The potential rate of change for software components vastly exceeds that for hardware components. This flexibility is a direct result of software's malleability; software is easier and cheaper to change, and it requires no retooling of production machinery."

An increased focus on software-driven systems-level interoperability may also explain the emergence of "reference architectures" as a deliverable from consortia. For example, the OpenFog Consortium recently produced a reference architecture document designed to address "the need for an interoperable end-to-end data connectivity solution along the cloud-to-things continuum." Much of this envisioned architecture relies on software to enable interoperable connections between system-level components.

Even organizations that have historically focused on more traditional specifications increasingly are recognizing the importance of software. The Internet Engineering Task Force (IETF) was a pioneer in this area, as their specification development process has long required concrete examples of "running code." More recently, organizations like the Broadband Forum, MIPI Alliance and many other consortia have implemented policies and practices aimed at incorporating software into the specification development and implementation process.

One challenge associated with the use of open source software code as a tool for interoperability is that it is effective at the moment in time when the relevant industry stakeholders have agreed to implement the shared code, but interoperability is potentially inhibited if any party diverges from that code. This challenge is exacerbated by the fact that the ability to diverge is an express feature of the open source license model. By definition, open source code is licensed in a manner that permits parties to make changes to the code. When one party unilaterally changes code that underpins an interoperable system, however, interoperability can break.

Consortia have been developing solutions to address this problem. Some organizations have been developing software code first, closely followed by a specification that defines a canonical implementation of that code. IoTivity and the Open Connectivity Foundation follow this model, with the IoTivity project developing code, and the OCF creating a specification. An organization called Alljoyn had previously followed this model as well, tying certain trademark and patent licenses to use of the canonical specified version of the code in an effort to incentivize ongoing compliance with a standardized implementation.

The collision between the traditional hardware-focused specification development process and a software-centric approach to interoperability has resulted in a clash of development methodologies, intellectual property models, and ultimately of cultures. Traditional consortia often apply a disciplined, systems engineering style approach to development. Open source projects sometimes embody a development methodology caricatured as "move fast and break things," but perhaps more fairly characterized by the IETF slogan "rough consensus and running code." Further, the FRAND intellectual property model used by some consortia can clash with the common expectation of the open source community that deliverables will be implementable on a royalty free basis.

Some leading consortia increasingly appear to be adopting more software-like methodologies, even when creating traditional specifications. The Khronos Group, for example, makes its specifications available and manages inputs via the popular software repository tool called GitHub. The World Wide Web Consortium similarly uses GitHub as a development forum, and increasingly its specifications themselves blur the line between traditional textual specifications and software code.

As "software eats the world," one part of the world that it appears to be eating is traditional consortia. Increasingly consortia produce software code deliverables, and software-oriented deliverables such as reference architectures. Further, the development methodologies, intellectual property models, and ultimately the culture of open source appear increasingly important in the development of interoperability solutions in the ICT industry.

C. The Remarkable Growth of Linux Foundation Illustrates These Structural and Substantive Changes

This Section III of this paper has made two main arguments: (1) the structure of consortia have evolved, and continue to evolve, to address a complex set of legal, tax and operational issues, and (2) open source software has increasingly become a key tool for ICT interoperability. The extraordinary growth of Linux Foundation serves as evidence in support of both of these points.

In 2013 Linux Foundation hosted 10 projects, including Linux itself. In early 2019 it hosts 156. In comparison, ISTO has supported about 50 projects in its 20 year history, and currently supports 17. VTM Group, a leading provider of support services to independent contractual and incorporated consortia founded in the late 1990s, lists 87 past and current clients on its website. LF's 150+

projects is thus a striking number.

By 2013 LF's revenues were already on a steep upward curve that appears to have begun in about 2010. 2013 revenues were over US$23 million. Four years later revenue had nearly quadrupled, to US$81 million. In 2013 LF reported 39 employees; in 2017 it reported 178. This author maintains a database of 132 U.S.-based consortia and standards setting organizations that are tax exempt under Section 501(c)(6) of the U.S. tax code, and compiles information reported on each organizations' IRS tax forms. No other organization comes even remotely close to Linux Foundation's growth rate, either in absolute dollars, percentage growth, or growth in human resources. LF's growth is extraordinary.

From a structural perspective, Linux Foundation appears to be delivering the set of legal, tax and operational solutions that meets the ICT industry demands. While it is perhaps too early to state definitively, the 'Series LLC' legal model pioneered initially by JDF and adopted by LF appears to an evolutionary legal innovation related to consortia that is sticking. More substantively, it appears that Linux Foundation's roots as a software development organization are right for an era when software is increasingly the tool of choice for driving interoperability. At the same time, Linux Foundation appears to be driving a synthesis of the software and specification-oriented models. Back in 2013 described that LF collaborative projects must meet two criteria: "the use of open source governance best practices including license and contribution agreement choices in keeping with the ideals of Linux" and "the project must have the potential to fuel innovation in an industry through collaborative software development." Contrast that to the description of a recently-announced "umbrella project," LF Edge, that combines several LF projects to target a sophisticated vision for systems-level interoperability leveraging both specifications (standards) and code (the LF website notes "the ultimate output being working code"):

> *LF Edge will create a common framework for hardware and software standards and best practices critical to sustaining current and future generations of IoT and edge devices. We are fostering collaboration and innovation across the multiple industries including industrial manufacturing, cities and government, energy, transportation, retail, home and building automation, automotive, logistics and health care – all of which stand to be transformed by edge computing.*

This paper has argued that interoperability is a fundamental ingredient to ICT industry success, and that consortia play a critical role in facilitating interoperability. It has suggested that consortia have evolved to address a complex array of legal, tax and operational issues, and are in the midst of a particularly acute moment of evolution as "software eats the world" and software increasingly becomes a key tool for interoperability. While Linux Foundation is far from the only player in the game, it has ridden these evolutionary trends in a remarkable, unique way. Any stakeholders considering the future of ICT standardization must consider the example of Linux Foundation and its increasingly outsized status as an industry leader.

IV. Europe in a New Era for Consortia

So, what does this all mean for Europe? One answer conceivably could be: nothing. That is, European companies and other European stakeholders are already deeply involved in the consortia and related processes that are described in this paper. Representatives from European companies participate in the leadership of nearly every major consortium. Some of the U.S.-based consortia described in this paper are primarily led by European interests, such as the LoRa Alliance and the OPC Foundation. Further, many consortia have deep, complementary relationships with European-based organizations like ISO, IEC and the ITU. One conclusion might be that the status quo is working for European stakeholders.

At least for the European Commission, however doing nothing is not the plan. In several EC communications (European Commission, 2016a, 2016b), and in the study proposal referenced earlier, the EC has stated a clear goal to "accelerate the development of common standards and interoperable solutions" as part of its Digital Single Market strategy.

This paper concludes by drawing from the points made earlier to offer several observations that may be relevant to the EC and other European stakeholders as they pursue this goal. It highlights some particular areas where the EC may be able to make unique contributions in this evolving environment of interoperability needs.

Some observations and recommendations:

1. Recognize the limitations of formal standards development organizations. Thirty years of experience shows that formal standards development organizations have not been able to comprehensively meet the interoperability requirements of the ICT industry. Consortia have grown increasingly important over time. This trend is unlikely to reverse. Europe has strong formal standards organizations, and a desire to rely on these organizations to effectuate a standards strategy would be understandable. Experience shows it will not work.

2. Investigate why few private sector-led consortia have formed in Europe. Relatively few consortia have formed in Europe. Some potential explanations include:

- Corporate liability risks or corporate formation complexity. AUTOSAR is commonly identified as a leading example of a Europe-based consortium in the genre of USB, WiFi or Bluetooth. AUTOSAR is a "BGB-Gesellschaft" or "GbR" entity under German law, generally the equivalent of a U.S. law partnership. Members of AUTOSAR are thus generally jointly and severally liable for acts of the entity. This approach reflects a disfavored model for U.S.-based consortia, which typically form using structural models that insulate participants from liability. Further, the emerging 'series LLC' model in the U.S. appears to achieve this result (plus other benefits) with little cost or administrative overhead. Law in various European jurisdictions may not offer the same combination of formation simplicity and risk reduction as U.S. law.

- Antitrust/competition law risks. The U.S.'s National Cooperative Research and Production Act and the related Standards Development Organization Advancement Act created a favorable legal environment for consortia in connection with antitrust risks, and in the ensuing years antitrust enforcers have rarely intervened in consortia activity. It's unclear whether European authorities would take the same consistently tolerant approach.

- Tax considerations. U.S.-based consortia typically form under a model that permits them to pay no federal or state income taxes. Accordingly, member fees paid to an organization, and other program service revenue generated by an organization, generally go 100% towards the activities desired by the members. The tax environment in Europe may be less favorable.

- Support infrastructure: operational support; tax, finance and legal advisors. Consortia require a broad range a support services: operations management, event planning, marketing expertise, various technical services, bookkeeping and auditing, tax compliance, legal support, etc. Organizations like VTM Group and their handful of competitors specialize in offering support packages for independent technology consortia. Organizations like ISTO and Linux Foundation are similarly dedicated to providing support infrastructure for their projects, and to making the formation process simple. Further, specialized law firms and other professionals focus exclusively on the unique needs of technology consortia. This sort of well-refined support infrastructure, with a highly-specialized focus on technology consortia, may be less robust in Europe.

3. Understand the critical role of open source software – including its royalty free license models and its culture. Software is increasingly fundamental to the creation of interoperable systems. Open source software development methodologies are different from traditional consortia models, and certainly very different from the standards development methodologies practiced by the formal European standards organizations. Open source developers will not see a need to change their processes to accommodate historic standardization approaches. Further, they will see little value in trying to impose FRAND licensing models in an environment where royalty free models have been demonstrably effective. If European stakeholders want to benefit from the capabilities of open source software, they must meet the open source community on the community's own terms.

4. Selectively embrace the role of 'convener' to break logjams. Consortia have been successful in part because they compete with each other. This process has been effective, and policymakers and regulators should not be quick to intervene in this competition. At times, however, consortia with competing visions can deadlock. For example, while its likely too soon to tell if there is a deadlock, the Open Fog Consortium and its allies, and the various constituent groups of the LF Edge program, have developed competing reference implementations for edge computing. The EC and particularly its partner institutions may be uniquely positioned to convene stakeholders and facilitate discussions that would not happen – or may only happen slowly – if left to actors driven exclusively by pecuniary motives. Convening stakeholders, but letting market forces ultimately drive decisions, could be a helpful role.

5. Selectively embrace the role of 'convener' to identify cross-industry systems architecture needs. The EC documents referenced above rightly emphasize the increasing importance of "reference architectures," particularly in the context of the broad systems-level changes implicated by the Marc Andreesen article. The EC and its partners may similarly be uniquely positioned to be able to convene cross-industry stakeholders to address systems level requirements in a manner that may be difficult for private sector actors. Consideration of the global context of many industries will be critical in this context: narrowly focusing only on European interests may inhibit long term success.

6. Continue to focus on testbeds and related compliance services. As described in the referenced communications, the EC has made considerable investments in "testbeds" that enable various parties to inexpensively test the practical interoperability of their products and services in real-world scenarios. This is a smart approach. Practical, working-level interoperability is the ultimate goal of a standardization process, but the difficult work of accomplishing this is often under-resourced. It is particularly challenging for small and medium-sized enterprises (SMEs). Offering testbed services as a public resource is a clever and unique solution to an important problem. The EC could conceivably build global standardization leadership off of these resources alone. Adding related services, such as formal compliance testing, potentially coupled with compliance logo licensing, is also worthy of consideration.

V. Conclusion

As Marc Andreesen suggested, developments in ICT are now poised to transform industries far beyond ICT. Consortia are a large part of what brought us to this point, and they will continue to play a critical role as interoperability requirements grow in complexity. Consortia themselves have evolved over time, and are continuing to change. Open source software is increasingly part of how interoperability happens, and today's consortia reflect this. The emergence and rapid growth of the software-centric Linux Foundation is striking evidence of this new reality. This story holds important lessons for European stakeholders. Within this changed landscape there are opportunities for Europe to play a unique, globally-leading role.

About the author

***C. Bradford Biddle** is a Faculty Fellow with the Center for Law, Science and Innovation at the Sandra Day O'Connor College of Law, Arizona State University, and the founder of Biddle Law PC, a boutique law firm based in Portland, Oregon that specializes in providing legal support for standards-setting organizations, open source foundations and other technology consortia. He thanks the organizers and attendees at the 24th EURAS Annual Standardisation Conference, held in Rome, Italy in June 2019, for their support and insightful comments about an earlier version of this article.*

References

Armstrong, A., Mueller, J. J., & Syrett, T. D. (2014, May 29). *The Smartphone Royalty Stack: Surveying Royalty Demands for the Components Within Modern Smartphones*. Retrieved from https://ssrn.com/abstract=2443848 or https://dx.doi.org/10.2139/ssrn.2443848

Andreessen, M. (2011, August 20). Why Software Is Eating The World. *The Wall Street Journal*. Retrieved from https://www.wsj.com/articles/SB10001424053111903480904576512250915629460

Baron, J. & Spulber, D. F. (2018, February 2). *Technology Standards and Standard Setting Organizations: Introduction to the Searle Center Database*. Northwestern Law & Econ Research Paper No. 17-16. Retrieved from https://ssrn.com/abstract=3073165 or https://dx.doi.org/10.2139/ssrn.3073165

Biddle, C. (2017). No Standard for Standards: Understanding the ICT Standards-Development Ecosystem. In J. L. Contreras (Ed.), *The Cambridge Handbook of Technical Standardization Law: Competition, Antitrust, and Patents* (pp. 17-28). Cambridge: Cambridge University Press.

Biddle, C., White, A., & Woods, S. (2010). How many standards in a laptop? (And other empirical questions). In *2010 ITU-T Kaleidoscope: Beyond the Internet?: Innovations for Future Networks and Services* (pp. 123-30). Pune: ITU. Retrieved from https://ieeexplore.ieee.org/document/5682128

Bluetooth SIG, Inc. v. United States. (2010, July 8). *Law.com*. Retrieved from https://www.law.com/almID/1202463375482/?slreturn=20190309032730

Cargill, C. F. (1989). *Information Technology Standardization: Theory, Process, and Organizations*. Newton, MA: Digital Press.

Carney, D. J., Fisher, D., Morris, E. J. & Place P. R. (2005). *Some Current Approaches to Interoperability*. Retrieved from https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=7511

CBR Staff Writer. (1989, August 21). Corporation for Open Systems Fires Half Its Staff. *Computer Business Review*. Retrieved from https://www.cbronline.com/news/corporation_for_open_systems_fires_half_its_staff/

European Commission. (2016a). *ICT Standardisation Priorities for the Digital Single Market* (COM(2016) 176 final). Brussels: The European Economic and Social Committee and the Committee of the Regions. Retrieved from https://ec.europa.eu/digital-single-market/en/news/communication-ict-standardisation-priorities-digital-single-market

European Commission. (2016b). *Digitising European Industry Reaping the Full Benefits of a Digital Single Market* (COM(2016) 180 final). Brussels: The European Economic and Social Committee and the Committee of the Regions. Retrieved from https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52016DC0180

European Commission. (2018). Study on Technological and Economic Analysis of Industry Agreements in Current and Future Digital Value Chains (SMART 2018/0003).

Hamilton Jr., J. A. & Murtagh, J. L. (2000). *Enabling Interoperability Via Software Architecture*. Retrieved from https://apps.dtic.mil/dtic/tr/fulltext/u2/a458021.pdf

Huawei. (2018). *Tap Into New Growth With Intelligent Connectivity: Mapping your transformation into a digital economy with GCI 2018*. Retrieved from https://www.huawei.com/minisite/gci/assets/files/gci_2018_whitepaper_en.pdf?v=20180914

Mähönen, P. (1999). The Standardization Process in IT – Too Slow or Too Fast? In K. Jakobs (Ed.), *Information Technology Standards and Standardization: A Global Perspective* (pp. 35-47). Hershey, PA: IGI Global.

NV. (2014, July 14). In Praise of the Humble USB. *The Economist*. Retrieved from https://www.economist.com/babbage/2014/07/14/in-praise-of-the-humble-usb

Open Source Compliance for Embedded Systems: a Closer Look at the GPL-2.0 and its Requirements

Dr. Hendrik Schöttle [a]

(a) Rechtsanwalt and Partner at Osborne Clarke, Munich, Germany

DOI: 10.5033/jolts.v11i1.136

Abstract
Open source compliance has achieved the next level: In the beginning of open source compliance actions, violations of open source licenses have been enforced by the community as a matter of principle. At the same time, awareness about compliance requirements have also risen, especially when distributing open source software as part of embedded systems. But is compliance nowadays more difficult than ever? And how does this apply especially for embedded systems?
Thoughts of a German qualified lawyer who has been involved in several compliance cases with a focus on the developments concerning GPL-2.0 compliance in Germany.

Keywords
Law; information technology; Free and Open Source Software; Open Source Compliance; GPL-2.0; GPL; Embedded Systems

1. Open Source Licenses as Playground for Trolls

Anyone who distributes open source software (OSS) as part of a product has to comply with its license conditions. Even though many companies avoid OSS, and even though in many cases I still see it wrongly assumed that no OSS is used: Today, most players dealing with OSS have realized that there are licensing conditions which entail special obligations and that these obligations generally have to be fulfilled.

In many cases, however, I still see helplessness with regard to which obligations have to be fulfilled in detail and what this means from a practical point of view.

This would not be an issue, as in most cases OSS communities and representatives of proprietary software get along better than just a few years ago. Many companies have realized that participating in OSS projects leads to a competitive advantage. OSS projects are no longer forked by companies, i.e. they are no longer developed independently from the original OSS project. Many companies give improvements, patches and bug fixes back to the community. Business models around OSS have also changed: A rising number of projects does not focus on the principle of free software anymore; many pursue quite openly economic interests - however, not with the sale of software, but with the services behind it.

Nevertheless, at least in Germany the number of cases, in which missing OSS compliance is enforced, is increasing. Initially, such enforcement of OSS license violations was driven by idealistic motives. Nowadays, it is often about financial interests of individuals. The majority of cases do not end up in court, but are settled outside of court.

Although many companies do know that OSS compliance is important, organizations and individuals when using OSS have problems to understand and to implement license compliance. Why is that? The main problem is likely to be the OSS licenses themselves. Many of them were drafted in the 1990's, some even in the 1980's. Even if the main principles of programming have remained the same, software development and its distribution have changed considerably over the years.

As a result, today we are dealing with OSS licenses whose obligations require expert knowledge and support in order to be fulfilled in practice with reasonable efforts. Ultimately, any company that distributes OSS components is thus exposed to potential injunctive relief from the copyright holders. This is repeatedly countered by OSS communities, who argue that in practice there is no need to fear legal disputes, as everyone gets along well with each other. However, this is only small comfort for those over whom the sword of Damocles is hanging. From a legal risk point of view, a single developer of the referring OSS is able to use injunctive relief against companies, prohibiting further distribution of their products with immediate effect, simply for violating formal OSS license obligations. The other developers' assurance that they will not take any legal action is irrelevant.

Let me get to the point: Many open source licenses used in practice have a potential for "trolling". They require compliance with conditions that can simply no longer be met in times of nowadays' software development with reasonable efforts - and thus offer the same opportunities that "patent trolls" do have. At least in Germany, the OSS community is now threatened with the same phenomenon that it has been fighting for years: the prosecution of violations of the law for commercial interests. This time, however, not by means of copyright and proprietary license agreements, but by means of OSS licenses.

In the following, the example of software which is licensed under the GNU General Public License, Version 2.0 (GPL-2.0), and used in embedded systems will show why the implementation of information obligations can be difficult in single cases. An Internet router may serve as an example. Its operating system is based on the Linux kernel, but unlike a classic desktop computer, the user has only limited access to the operating system. Of course, there is not only the GPL-2.0 as open source license – many more different licenses do exist. However, as the GPL-2.0 is one of the licenses with strict obligations and as it is enforced in practice repeatedly, it may serve as a good example.

2. The Story so Far

2.1. OSS Compliance 1.0

A little more than ten years ago, legal experts in Germany still disagreed as to whether OSS licenses were legally enforceable at all. Licenses such as the GPL provide for a transfer of rights which is conditioned by the fulfilment of several obligations - a legal construct which at first sight runs against many legal systems such as those established by the German Copyright Act.

In 2004, the GPL-2.0 was tested in court for the first time. The Munich Regional Court did not only decide that the construct of the conditional transfer of rights is effective and does not constitute an unreasonable disadvantage in the sense of the German law on standard terms and conditions (which would have led to its unenforceability).[1] The court also held that a missing license text and missing source code is a breach of the GPL-2.0's obligations, which can lead to injunctive relief claims.

In the period that followed, missing license texts and missing source codes of GPL-licensed components were repeatedly enforced before court. Many of these enforcements were driven by idealistic motives: In the course of increasing software modularization, more and more OSS components were used in commercial products, but not declared as such. After an initial phase, where OSS was not taken seriously, it soon conquered the world and found its way into commercial products. The community wanted to be taken seriously and this included fulfilling the OSS licenses' requirements.

2.2. OSS Compliance 2.0

In the initial phase, the dispute over compliance with OSS licenses was mainly about missing license texts and source codes, but soon it was about more: In 2012, the Bochum Regional Court ruled in a landmark decision that even if OSS was basically free of charge, damage claims were possible according to the principles of license analogy.[2] This answered the question of whether it was just a matter of principle or money.

Moreover, compliance was about more obligations than providing the license texts. Suddenly it was also about missing copyright notices, about a missing or insufficient offer of the source code or about a missing disclaimer - obligations, whose compliance even large parts of the OSS community had not taken seriously for quite a while.

Not surprisingly, there are now disputes within the OSS community about how to properly deal with license violations. Only recently, a developer was expelled from a development team; he was blamed of pursuing his own financial interests.[3]

Additionally, disputes over OSS license infringements are no longer limited to the classic relationship in which a right holder of OSS licenses takes action against an infringer. Nowadays, there are also disputes between companies about the effect of OSS components used, in the development of which the companies were not directly involved. For example, in the *Versata vs. Ameriprise* case before a US court, a software vendor and its customer disputed whether the copyleft effect of a GPL-licensed OSS component results in the proprietary software distributed by the software vendor being able to be used as OSS free of charge.[4] This further increases the risk of being taken to court for license violations.

3. The Obligations of the GPL in Detail

3.1. Delivery of the License Text

First of all, the GPL requires that a copy of the license text be provided with the licensed software. Specifically, the GPL-2.0 requires the following in paragraph 1:

1 Regional Court of Muenchen, Judgement of May 19th 2004 – File number 21 O 6123/04.
2 Regional Court of Bochum, Judgement of March 3rd 2016 – File number I-8 O 294/15. In case of a license analogy, damage claims are calculated on the basis of a fictitious or existing license fee that could be charged for legally compliant licensing, usually added by an additional surcharge.
3 *H. Meeker*, Patrick McHardy and copyright profiteering, accessed at https://opensource.com/article/17/8/patrick-mchardy-and-copyright-profiteering
4 United States District Court of Travis County Texas, Judgement of August 13th 2008 Case No. A-14-CA-12-SS; Versata vs. Ameriprise.

> *"You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you […] give any other recipients of the Program a copy of this License along with the Program."*

The GPL-2.0 dates back to 1991 - so let's go back to a time when software was distributed in cardboard boxes on 5.25" floppy disks together with a printed user manual. Software did not consist of too many sub-components, which made it feasible to list all of them in the user manual together with a handful of copyright notices. In this case, a copy of the license text could easily have been included in the manual to fulfill the obligations.

Today, the situation has changed. Software packages have grown from a few megabytes to gigabytes in many cases. Even smaller software contains hundreds of sub-components while in the past in many cases storage space was the limiting factor. Additionally, software is in most cases distributed without a physical copy. Even though boxed software still exists, it has been outpaced by downloaded software by far. As a consequence, the number of different license texts applicable to software included in a product has grown as well.

What about embedded systems? Let's think about the Internet router mentioned above. Instead of a manual, there might be a leaflet in the package that contains a short manual of a few lines of text and refers to a website for further information. How can the obligation to pass on the license text be fulfilled?

A reference to a website is not sufficient - the wording "give [...] a copy of this license along with the program" is clear in this respect; by the way, this has also been confirmed by the Munich Regional Court.[5]

It may be possible to include the license text in paper form in case of one single license. However, today many products contain numerous third-party components under dozens of different licenses, each of which requires the distribution of the license text. This quickly leads to a document that can contain several hundred pages of text consisting of the numerous different license texts and copyright notices (see below 3.3) contained in the product. Especially in the case of low-priced products produced at high quantities, the delivery of an additional paper copy can make the distribution of the product considerably more expensive - or make it economically impossible in comparison to competing products.

An alternative may be to show the license text on the display of the device - if the device has one. However, it becomes difficult if such a display is missing. The Free Software Foundation, which was involved in the drafting of the GPL, is of the opinion that it is sufficient to reproduce the license text via the interface used by the device. In its FAQ, the Free Software Foundation wrote the following:

> *"Q: My program has interactive user interfaces that are non-visual in nature. How can I comply with the Appropriate Legal Notices requirement in GPLv3?*
>
> *A: All you need to do is ensure that the Appropriate Legal Notices are readily available to the user in your interface. For example, if you have written an audio interface, you could include a command that reads the notices aloud.*[6]

In the opinion of the Free Software Foundation, it would therefore be sufficient if, for example, a headset reads the license text aloud. Even though the Free Software Foundation is definitely an institution that cannot be underestimated in questions relating to the GPL, one may also have a different

5 Regional Court of Muenchen I, Judgement of May 12th 2007 – File number 7 O 5245/07.
6 Frequently Asked Questions about the GNU Licenses – https://www.gnu.org/licenses/gpl-faq.html#NonvisualLegalNotices. This FAQ refers to the GPL-3.0. However, as Clause 4 GPL-3.0 is identical with Clause 1 GPL-2.0, we see strong arguments that the FSF's statement can be applied accordingly to the GPL-2.0.

view on this. It can be doubted whether a voice recording can be regarded as a provision of a license text, leaving alone the comprehensibility of the acoustic transmission of such texts.

If however, following the opinion of the Free Software Foundation, this is sufficient, then other interfaces could very likely also be used. For example, a device is connected with a local network by the user by connecting with an external device such as a smartphone or notebook computer to a Wi-Fi hotspot set up by the device. In a second step, the user then enters the necessary data on a web server run on the device (as this is the case with almost all Internet routers). In this case, it should be sufficient to store the license text locally on this web server within the embedded system. This would not qualify as a link to an external source in this case, the license text would rather be stored on the same storage medium as the software.

Another solution may be the provision of a CD-ROM or a DVD containing the necessary information. However, in this case, it has to be considered whether the recipient will be able to read data from such CD-ROM or DVD. This is another good example for changed requirements due to technical developments: While 15 years ago, in private households, an Internet router will very likely have been the gateway between the Internet and a desktop computer, in most cases equipped with a CD drive, nowadays many households are only using smartphones or tablet computers only and do not have the possibility to read CD-ROMs or DVDs any more. As a result, such provision may be sufficient in a B2B scenario where such equipment is more likely to be present at the recipient or in a B2C scenario in cases where the existence of a respective drive is likely.

In theory, the above solutions sound quite simple. In practice, however, considerable efforts must be invested to find the required license information. Software components that are integrated into the own software often contain further subcomponents, which again contain subcomponents, and so on. The Linux kernel alone brings up more than 60,000 files, in which hundreds of license texts are hidden, which are available in various formats. Furthermore, files can have multiple licensing statements. There are scanning tools which are used by many organizations in the context of compliance processes and which search the source code for license texts or single copyright notices. However, persons must be aware that in some cases, experts need to correct the results generated by these tools. Standardization of how to express license and copyright information is still emerging; as such, many OSS today did not implement it so far[7]. Anyone who wants to ensure that any and all license texts have been captured correctly will have to depend on manually reworking the scan result.

Additionally, as even complex software has sneaked into everyday products, license requirements may have to be fulfilled where a distributor does not even think of software being contained in a product.

Thus, license compliance is not that easy, as software is nowadays distributed in many more ways than about 30 years ago and as software contains much more sub-components than at that time. As the hardware has become cheaper as well, it can become an economical hurdle to provide written documentation together with a low-priced product.

3.2. Reproduction of Copyright Notices

In addition to the delivery of the license text, GPL-2.0 also requires the reproduction of copyright notices. Clause 1 of the GPL-2.0 states with this regard:

> *"You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; […]."*

7 See e.g. the FSFE's reuse.software project at https://reuse.software

Even though the handling of copyright notices under GPL-2.0 seems like a classical obligation, a closer look at the license texts shows that its requirements are far from being clear:

It is already unclear, whether the distribution of software requires to attach a copyright notice, even when distributing unmodified software or whether it is sufficient to retain existing copyright notices. The language of Clause 1 states that one has to "*publish on each copy an appropriate copyright*". Taking this requirement literally, one would have to add a copyright notice in any case, even when not modifying the software. Such interpretation could be backed with regard to the following clause of the GPL-2.0, which requires to "*keep intact all notices that refer to this License*": Obviously, the GPL-2.0 differentiates between the "*publication*" of a notice and "*keeping a notice intact*". However, this can hardly be meant as a requirement when distributing unmodified software. In such a case, the distributor cannot be regarded as copyright holder of the software, as no changes would be made to the existing code. So in this case, it would not only be unnecessary to add one's own copyright notice, but it would also be misleading. However, one could still interpret this clause requiring to add a copyright notice of upstream copyright holders. In this case, a distributor would have to do research on possible upstream copyright holders and complete missing information. However, even though such interpretation could be backed by the language of Clause 1 as well, it would be error prone. Thus, in German literature, experts advise not to add any copyright notices to unchanged code, they hold that it is sufficient to reproduce existing copyright notices.[8] This seems to be the most convincing result and also what is seen as common practice, even though it may be difficult to overcome the license's language which is stricter, as it clearly differentiates between "*publishing*" and "*keeping intact*" notices.

The vast majority of copyright notices can be found in the individual source files. The single programmers include a note concerning their contributions there in form of a comment. However, these comments are removed when compiling the software into binary code. The binary code usually only contains a general copyright notice in the software, which is displayed when the software is executed. All notices contained in the source code are missing.

Firstly, it is almost impossible to compile the copyright notices when passing on OSS in binary form only. Why is this so? So is it necessary to extract such comments from the source code and to compile them in a separate document in order to ensure their delivery with the binary code? Let us take a closer look at the license: The Problem is the reference chain from Clause 3 to Clause 1. Clause 1 deals with the distribution of the source code. When it comes to distribution of the binary code as described in Clause 3, this Clause simply refers to Clause 1:

> *"You may [...] distribute the Program [...] in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following [...]"*

Following this referral to Section 1, when distributing the software in object code, one again would have to "publish" a copyright notice. As outlined above, it is probably sufficient to retain existing copyright notices, even though the language of Clause 1 suggests more. However, as during the compilation process, the copyright notices are removed from the code, one may very well argue that they have to be reinstated, especially as Clause 1 speaks of "publishing" such notices, not of "keeping them intact". According to German legal experts, the requirement to extract copyright notices from the source files can be backed by a strict interpretation.[9] Other voices point out that increasingly more copyright holders insist of exactly doing that.[10]

8 O'Reilly, Die GPL kommentiert und erklärt, marginal 32.

9 *Jaeger/Metzger*, Open Source Software, 4th Ed. 2016, marginal 37.

10 *Hemel*, Practical GPL Compliance, p. 39.

Thus, there is a risk that a copyright holder may require said extraction of copyright notices from the source files when distributing the binary, even though such extraction does not seem to be common practice.

So if you follow the strict approach, it would be necessary to collect the copyright notes from the source files and merge them into a separate document. If you have several software components, you also have to assign the notes to individual files additionally. With regard to the Linux kernel already mentioned, this can sum up easily to several thousands of source files, which then have to be searched individually for copyright notes. By the way, this obligation applies regardless of any modifications to the software. Thus, anyone who distributes an unmodified Linux distribution in binary code must go through this exercise, unless the necessary information has already been provided. Even though some scanning tools are specialized in collecting such copyright notices, naturally they do not find everything and require manual corrections or reviews.

Additionally, the formal requirements for copyright notices are stricter than for the delivery of the license text. The Free Software Foundation refers to the fact that it is generally required to deliver the source code anyway and that a delivery without source code is only permitted in limited exceptions.[11] Since the copyright notices are contained in the source code, isn't it enough to simply deliver the source code with the product in order to fulfill the requirements? Well, paragraph 1 of the GPL-2.0 requires that “an appropriate copyright notice” be published, namely “conspicuously and appropriately”. So a conspicuous and appropriate publication of a copyright notice is required, which again has to be appropriate.

Let's go back in time to the early nineties: Software is distributed on floppy disks or floppy disk images shared via the Usenet. The software user inevitably has access to a floppy disk drive and a desktop computer - because the software cannot be used in any other way. So if the source code is also delivered with the binary code, the user can read it with a simple text editor and has the possibility to view all copyright clauses. Accordingly, legal experts hold it as sufficient to deliver the copyright notices together with the software.[12]

So can the source code of embedded systems simply be stored together with the binary files in the memory of the device in order to fulfill the information obligations? This case is somewhat different from the desktop computer: With most embedded devices, the user does not have full access to the file system. The user cannot access and read source files stored on the embedded device when using only the means of the embedded device. The information may be stored on the device. Whether it is “appropriate” may be doubted, but they can probably not be considered as catching the eye in the sense of the term “conspicuously”. As outlined above in section 3.1, it depends on the individual case whether it is sufficient to include a data carrier such as a DVD or a flash drive. It may be compliant if it is sufficiently certain that the typical user of the embedded device has an appropriate reader, i.e. a desktop computer or a notebook with USB ports. In many cases, however, one may not this will no longer be mandatory.

The Free Software Foundation however even goes one step further. In response to this article’s author's question, whether the copyright clauses had to be extracted manually from the source files if only the binary files were distributed, they replied:

> *“We would describe a written offer as distributing the source […] Delivery of binaries with just a written offer […] is perfectly acceptable. […] I’m also not aware of someone putting in place the practice you describe.”* [13]

11 https://www.gnu.org/licenses/gpl-faq.html#UnchangedJustBinary .
12 *Koglin*, ifrOSS, Die GPL kommentiert und erklärt, 2005, p. 50 marginal 36.
13 Quote from an e-mail of the Free Software Foundation dated August 22nd 2016.

Thus, according to the Free Software Foundation, it should even suffice if the copyright notices are not provided at all, but if only an offer is made to receive the source code. This would then be equivalent to the delivery of the source code and thus the information obligations would also be fulfilled.

The GPL does indeed allow a written offer addressed to the general public to receive the source code instead of its distribution. However, this only applies to the source code and not to the copyright notices. The Free Software Foundation may take a liberal view here - but the wording of the GPL-2.0 alone does not reflect this view. Even if there are good arguments for such an interpretation of the GPL-2.0, an uncertainty remains.

Does the wording of the GPL-2.0 still matter now, where the Free Software Foundation obviously takes a different view? In the OSS community, the Free Software Foundation is often compared with a legislator, since it contributed significantly to the wording of the GPL and thus also claims the sovereignty to interpret it. The view of the Free Software Foundation may be an important one, but it is not necessarily the authoritative one. The GPL is not a law, but a template. And if this template is used between two contracting parties, at least under German law, any ambiguities concerning its interpretation are resolved by interpreting firstly the wording of the text, secondly by the underlying understanding of the parties and only in last instance by referring to possible legal opinions of third parties.

There are good arguments that one may interpret the GPL's requirements in accordance with the Free Software Foundation restrictively and come to the result that, despite the opposite wording, due to the changed form of the software distribution a restriction of the information duties is required, so that with the distribution of the binary files no copyright notices must be compiled manually from source files and be delivered with the binaries. Even if there are no references to such an interpretation in literature and case law, one still has the Free Software Foundation on its side.

3.3. Reproduction of a warranty disclaimer

Besides the copyright notices, Clause 1 of the GPL-2.0 also requires the provision of a warranty disclaimer. And again, even though this obligation as well seems to be pretty standard, like the copyright notices, it shows an inconsistency that leads to uncertainty and possible compliance issues in practice.

As outlined above, Clause 1 of the GPL-2.0 requires to publish on each copy a warranty disclaimer. This obligation makes sense in case of source code provision – and it becomes clear when taking a look at the Section "How to Apply These Terms to Your New Programs", which is not part of the GPL-2.0's license text itself. With regard to the notices, including the warranty disclaimer, this section recommends "*to attach them to the start of each source file to most effectively convey the exclusion of warranty*". Such procedure makes sense in order to ensure an increased level of comfort regarding the warranty exclusion – even though strictly speaking such additional disclaimer would not be necessary, as the GPL-2.0 as such already contains a warranty disclaimer in Clauses 11 and 12.

However, due to the reference chain from Clause 3 to Clause 1 in case of binary distribution, the obligation concerning the warranty disclaimer does not make sense any more: There is no source code, in whose header section a warranty disclaimer could be integrated. However, the GPL-2.0 still requires its provision in addition to the warranty disclaimer which is already contained in the license. As a result, a distributor of binary code has to provide the license text of the GPL-2.0 and an additional warranty disclaimer. As such disclaimer, one may use a verbatim copy of the boilerplate example of a warranty disclaimer which can be found at the end of GPL-2.0's license text. Even more confusing: the provision of the complete license text together with this boilerplate disclaimer would not be sufficient – as the warranty disclaimer is only an example, but not a disclaimer.[14]

14 The GPL-2.0 only suggests to "attach the following notices to the program", but it does not attach this sample text as actual disclaimer.

As a result, when distributing GPL-2.0'ed binary code, one has to add an additional warranty disclaimer at the end of the license text or elsewhere, in order to comply with formal requirements of GPL-2.0, even though it is hard to see an added value in providing this additional disclaimer.

This is by the way an issue that is not properly handled in many open source projects as well. Even the Linux kernel does not contain an additional warranty disclaimer. Such disclaimers can be found in single source files of the Linux kernel here and there – but not in all of them. As there is no general *additional* warranty disclaimer on a higher level in the Linux kernel package,[15] even the distribution of the unmodified kernel source files would constitute an infringement of the GPL-2.0, as for those source files without an additional warranty disclaimer the requirements of the GPL-2.0 are not fulfilled. As outlined above, the disclaimer included in the license text as such is not sufficient.

I would be glad if I could state that this is only a theoretical issue when following a very strict interpretation of the GPL-2.0. However, I have advised cases in practice where distributors have been subject to copyright infringement claims based exactly this issue: a missing additional warranty disclaimer in case of distribution of GPL-2.0'ed software, even though the license text including a warranty disclaimer had been provided together with the software.

There are good arguments that an additional warranty disclaimer is not necessary with regard to the purpose of the GPL-2.0, as the disclaimer already included in the license text is sufficient – it has to be provided as part of the license text anyway. However, strictly following the literal interpretation of the license text, an additional warranty disclaimer cannot be avoided.

3.4. Availability of the source code

Finally, section 3 of the GPL-2.0 requires the availability of the source code. This obligation can either be fulfilled by delivering the source code directly with the product[16] or by making a written offer to give any third party the source code on a medium customarily used for software interchange in return for reimbursement of the costs of reproduction, such offer to be valid for a period of at least three years.[17] In case of a non-commercial distribution one also has the possibility to refer to a respective offer one has received,[18] however, this possibility is excluded in case of a commercial distribution.

In practice, this means either the delivery of the source code or the submission of a written offer. At least under the widespread GPL-2.0 it is not sufficient to refer to a download option, as rather the source code must be provided on a standard data carrier. Of course, it does not hurt to refer to a download option in addition, as this will be the standard way in practice. However, from a legal point of view, if one does not provide the source code directly with the binary, one cannot avoid such an offer. It should be noted that such an offer cannot be found in the text of the GP-2.0 itself – thus, it has to be added to the documentation when distributing the software.

The Free Software Foundation provides a template text of such an offer, which is limited to three years and in its opinion meets the license requirements.[19] However, when adopting this text, it should be checked whether the time limit of three years for the offer can be accepted in the specific case. Some other open source licenses contain similar obligations to make such an offer, but for an indefinite period.[20] Anyone who now accepts the time limit proposed by the Free Software Foundation

15 When taking a look at the latest stable release 5.5.1, downloaded from kernel.org at the time of publication of this article, the file *COPYING* only referred to the files *LICENSES/preferred/GPL-2.0*, *LICENSES/exceptions/Linux-syscall-note* and *Documentation/process/license-rules.rst*, none of them containing an additional warranty disclaimer.

16 Sec. 3 a) GPL-2.0.

17 Sec. 3 b) GPL-2.0.

18 Sec. 3 c) GPL-2.0.

19 https://www.softwarefreedom.org/resources/2014/SFLC-Guide_to_GPL_Compliance_2d_ed.html#appendix-1-offer-of-source-code

20 For example, see Sec. 3.1 of the CDDL-1.0/CDDL-1.1.

must ensure that it only applies to the GPL-licensed components and not to other licenses, as otherwise a violation of the obligation to submit an unlimited offer is given. In addition, one risks of violating the wording of the GPL-2.0 even if using the sample text if it is not clear when the three-year period begins.

If the source files are delivered on the embedded device - which should be fine at first sight - it must nevertheless be ensured that copyright notices and any warranty disclaimers contained in the source code, which must be made available in a conspicuous and appropriate manner, can also be perceived in such a conspicuous way, see 3.3 above.

4. Conclusion

As a result, when following a strict interpretation, the obligations of GPL-2.0 lead to relevant efforts for ensuring license compliance when interpreting it strictly – not only in case of embedded systems:

For example, anyone who distributes a Linux distribution in binary code without modifications, usually commits a license violation if the source code is missing. Following a strict interpretation of the GPL-2.0, the distribution of the binary code requires extraction of the copyright notices from the source code.

But even in case of distribution of the unmodified sources, in many cases a license violation would be given, as an additional warranty disclaimer is missing, as e.g. in case of the Linux kernel.

Even those who deliver the source code with an embedded system are at risk of not fulfilling some obligations. The copyright notices must be appropriate and conspicuous. Even if there are good arguments for a restrictive interpretation of the GPL-2.0: one may doubt that this obligation is fulfilled if the source code cannot be accessed by the user of the embedded system without additional means.

To boil it down, distribution of open source software, specifically of GPL-2.0 licensed software faces two important tasks, when interpreting the GPL-2.0 strictly: First of all, it is not possible to distribute software without adding additional information, be it missing copyright notices in case of binary distribution or a missing warranty disclaimer in most cases of software distribution, regardless of whether in source or in binary form.

Second, the GPL-2.0 is focused on source code distribution, which may have been the best choice in early days' distribution of desktop software and is still important to the open source community - but is something that the majority of end users of embedded systems does not care about. The typical user of an embedded system does not want to modify the firmware or compile it in order to get the embedded system running.

Of course, one may say that it is simply the main intention of the GPL-2.0 to distribute respectively licensed software in source code form and in binary form as an exception. Nonetheless, it would be helpful to set up clear guidelines for binary code distribution and to include all necessary license compliance information into the OSS distribution once from the start instead of requiring every single distributor of such software to complete and edit missing information.

For some few developers who have contributed to OSS, OSS compliance has proven to be a goldmine - to the chagrin of the community, which simply wants to develop good software, and to the chagrin of the industry, which faces requirements that are not easily met without dedicated effort and that are disproportionate to the benefits.

Many companies are burying their heads in the sand in view of the efforts entailed by compliance requirements and hope that they will not get hit.

From the perspective of the advising lawyer one may cynically conclude that everything is fine, the need for advice is sustainably secured for the near future. But it would hardly be in the interest of the involved stakeholders if the distribution of GPL-licensed software was to reach legal limits due to technical developments. The involved stakeholders should stick their heads together and find a solution that gives both sides legal security. The good thing is that this already happens with regard to certain projects.[21] Such efforts to clean up OSS projects should be intensified and cover more and more projects. After all, open source software should not serve individuals, but the community.

So what to do in practice in the meantime? Best practice in case of binary distribution of GPL-2.0 licensed software in embedded systems is probably the extraction of copyright notices from the sources, addition of a warranty disclaimer and provision of the written offer for the source code. In case the extraction of copyright notices is considered too burdensome, one may refer to the FSF's statement on the written offer of the sources containing the copyright being sufficient with this regard as well. Concerning the form of provision, the best and in most cases most expensive way would be a printed copy, delivered together with the product. A second-best solution would be accessibility on the product's electronic display, if any. A fallback solution may be the provision in form of documentation locally stored in an internal web server, accessible via a Wi-Fi hotspot. A mere reference to an external online resource is not sufficient, at least according to the license text of the GPL-2.0, which has been confirmed by German case-law. In any case, a software distributor will have to ensure by technical and organizational measures that the OSS licenses' requirements are complied with – be it manually or by relying on automated OSS toolchain solutions.

The world has changed since the 1990's and so have the technical requirements for software distribution. The distribution of software as part of today's embedded systems shows that problems and questions have emerged which have to be addressed. Hopefully, this article helps to identify some of them and to provide at least workarounds, if not fixes for them.About the authors

About the Author

Dr. Hendrik Schöttle*, Rechtsanwalt, Partner, Specialized Lawyer in IT law, is Partner in the Munich Office of Osborne Clarke in Germany. He advises on IT law and data protection law, with a focus on software licensing models, in particular relating to open source software. He is frequently named in lawyer rankings in the field of IT law; the German JUVE handbook 2019/2020 recommends him as "leading name in the area of open source". Hendrik became a lawyer in 2005 and joined the Munich office of Osborne Clarke in 2007. He spent a number of years as a software developer at the Institute for Legal Informatics of Saarland University. He is the author of numerous publications and co-author of several handbooks and commentaries on IT law related topics, a lecturer in IT law at the Deutsche Anwaltakademie (German Lawyers' Academy) and a frequent speaker on topics relating to IT law. Hendrik is a board member of the BITKOM working group Open Source, a member of the Data Protection Law Committee of the German Federal Bar Association, a member of the information technology working group of the German Lawyers Association (DAV), and a member of the German Society for Law and Computer Science. He can be reached at hendrik.schoettle@osborneclarke.com*

21 *Corbet*, SPDX identifiers in the kernel, accessed at https://lwn.net/Articles/739183/ for further reference.