Exploring Freedom
A Conversation between FLOSS-Culture and Theological Practices of Freedom

Benedikt Friedrich
Ruhr-Universität Bochum
benedikt.friedrich@rub.de

The idea of the 'Freedom of a Christian' entails concrete practices of freedom. In order to unveil this connection, this paper compares practices of the 'Free Software Movement' with key insights of the Reformation and with how Protestantism develops its theology.

Introduction

"If contemporary theology has a central theme at all, it is Christian freedom."[1] With this statement, the German theologian Eberhard Jüngel highlights the foundational role of the concept of 'freedom' in theological discourse. And while, forty years later, there are still numerous scholars who would agree with Jüngel's sentence, it is crucial to investigate which role *freedom* plays in dogmatic and ethical inquiry beyond claims of subjective independency and personal sovereignty.

In this paper, I want to examine how every conceptualization of freedom intertwines with the social practices and structures it entails. In order to do so I will describe the phenomenon of *Free Software* and the *Free-Libre-Open-Source-Software* (FLOSS)

[1] Jüngel (1978), 16.

Zeitschrift für explorative Theology
CZeTh_3 (2022) S. 47–70
DOI: https://doi.org/10.17885/heiup.czeth.2022.3.24526
Zuvor veröffentlicht auf PubPub: http://doi.org/10.21428/fb61f6aa.9cf71305
Open-Access-Lizenz CC-BY-SA 4.0

cultures it has shaped.[2] Irrespective of theology, the *Free Software Movement* has successfully developed ways to implement its distinct notion of *freedom* as a mode of cooperative engagement in the creative processes of software development.

After explaining the specific aspects of these practices and how freedom is performed within this field, I will examine resembling structures in the field of Protestant theology, where there exists an analogous intertwining between the notion of the 'Freedom of a Christian' and concrete theological practices of freedom that derive from this statement. I will examine the pneumatological implications on each level of these localisations, which will result in a freedom-based understanding of how theological knowledge is produced.

1. The Free Software License

What is today known as *Free Software* began with Richard Stallman's founding of the *GNU-project* and with his vision to create of a fully functional operating system without copyright restrictions. His main concerns were a participatory and innovative mode of software development as well as the granting of non-restrictive access to the very core of all computer programs: the source code.

By developing his idea of *Free Software* ("think of free speech, not free beer"[3]), Stallman introduced a concept of freedom that applied directly to the very practices of software developers. Of course, ultimately Stallman was not only concerned about the freedom of developers, but of everyone living in an information based society. In fact, the ideology that drives the *Free Software Movement* even today largely consist of a quasi-eschatological vision of commonly shared knowledge and of a just society free of restricted access to intellectual goods.

The *Free Software Movement* understood that, in order to address the freedom of humans, it was necessary to center the discussion around the involved medium, the source code. It is thus mainly concerned with the freedom of software, in which users participate through various modes of interaction.

[2] I use the term "Free Software" when I explicitly refer to the actual "Free Software Movement" founded by Richard Stallman as well as when I refer to the legalistic implementations of distinct *free* licenses that are compatible with the *General Public License (GPL)*. The term "Open Source Software" is sometimes used as a synonym but it implies a different policy because of the incorporation of another moral framework. Nevertheless, the concepts of *Free Software* and *Open Source Software* share many of their practices and habits. Thus, when I refer to the practices of *Free Software* I use the term *FLOSS* (free, libre, open source software), which serves as a general term of the phenomena that emerge out of *Free* and *Open Source* Software alike.

[3] The Free Software Foundation (FSF), What is free software? https://www.gnu.org/philosophy/free-sw.html.en.

But *Free Software* does not just approach freedom as a distinct practice of writing code; it also engages the question of how these practices can be sustained and structurally secured. By inventing and using 'free licenses' (such as the General Public License, GPU, in its several versions), the movement introduced a powerful instrument that promised to provide a legal framework both for the moral visions and the pragmatic dimensions of free software development. It is an ironic incident, then, that law-like licenses have become the very epitome of freedom of FLOSS.[4]

At the core of most free software licenses are four paragraphs that state the basic principles of free software:

1. Users can freely use the software for any purpose. This is the most essential statement of the license.
2. Users are free to examine and adapt the software to their own needs. This implies that free software is shipped as open source, in contrast to proprietary software distribution of binary code, which won't allow the user to study its inner mechanisms and the way it was conceptualized.
3. Users are not only allowed to customize but also to redistribute the software, with or without additional modifications.
4. While users can improve the software, extend its functions, and make it easier to use, they are obliged to share those modifications with the public. This of course also means that the source code of this redistributed new 'version' needs to be publicly available.

Although licences like this challenge many juridical systems with respect to intellectual property, they have achieved a major transition in the way authorship of software and other intellectual inventions is assigned: It is no longer a question of property but a question of engaging in a solution-seeking community.

The GPL has now been a reference license for free and open software for over thirty years, but it has also been at the core of great disputes among the community and sometimes is even seen as the dividing line between distinct ideological groups within FLOSS.

[4] For example, the GNU General Public License opens with highlighting its purpose as a counter-narrative to proprietary software licenses: "The license agreements of most software companies try to keep users at the mercy of those companies. By contrast, our General Public License is intended to guarantee your freedom to share and change free software" (GPL, Preamble).

Extremely fast-growing developer communities arose, driven by the goal of defeating *proprietary software*, which had become the industrial standard for at least two decades. Several operating systems (best known are GNU/Linux and BSD) were developed and licensed as free software. And despite the unquestionable success of Microsoft Windows and Apple OSX in desktop computing, a huge amount of *Free Software* projects have emerged, developing software for nearly every purpose. The fact that most of the internet's infrastructure, the implementations of several standards within communications technology, and the core (kernel) of every Android and iOS-device is nowadays based on *Free Software* shows that it has become much more than a small counter-movement within hacker communities. Indeed, by establishing complex practices of cooperative and decentralized work driven by this distinct vision of freedom, *Free Software* has made a deep impact on how our digital world is structured today.

It is crucial to understand that the transformative power of *Free Software* is not only measured by the extent to which its projects have spread. Beyond the undeniable success of the movement in that regard, it is also insightful to investigate the structural consequences and the practical implementation of its very vision of freedom. From a freedom-theoretical perspective, it is remarkable that *Free Software* has found ways to derive habitually and structurally formative practices from these ideological and legal foundations of freedom.

2. Recursive Publics and Their Platforms

Free Software licenses understand *freedom* as a mode of interacting through a certain medium: the medium of code. Freedom in FLOSS is therefore not a goal in itself but a mode of collaborative interaction that takes the structural and habitual necessities into account.[5] This can be illustrated by looking at several paradigmatic operational procedures. Free software not only gives anyone who chooses to engage in software development the chance to contribute their own ideas to a specific project, it also provides appropriate platforms and environments that allow developers to work on the same code cooperatively and even simultaneously. Developer communities have invented numerous tools for collective code manipulation for this purpose. The initiative for the creation of these tools came from the core of the *Free Software Movement* and was driven by the vision of sharing the different capabilities and the highly specific knowledge of a great number of people, required for new and original solutions. They

[5] Cf. Kelty (2008), 2.

understood that milestones in large scale projects can only be reached with a critical mass of participants.

The most important part of the decentralized development process are 'version control systems.'[6] They are tools that structure the way people collaborate in software development by displaying the process of a project's growth and thus making the history of co-authored development transparent.

Everyone who has the technical abilities of reading and writing code can engage in the improvement of different parts of a program by implementing new features, fixing bugs and security issues, adapting it to personal needs, or making it easier to use. They can either adjust a program to their own needs or engage in the development community if they consider their ideas beneficial for others.

Depending on the organizational structure of a given project, people can either directly submit their ideas of improvement (into the so called 'master branch') or they can submit their suggestions by creating their own new branch. This is the very point that decides whether the freedom of FLOSS leads to a fragmentation of different branches (where everyone starts their own branch) or to a culture of co-dependent joint development. The technical term for this process is 'pull-request': handing in a code snipped to the main version of given project. Pull-requests aim at solving existing security-issues and at finding solutions to both known and overlooked problems of a program. They eventually reveal new possibilities for improving the software. Pull-requests can also – in a non-deficient-oriented way – add one's own sense of creativity to the project, through contributing ideas of further development that entail new functions and directions that would meet the needs of other users.

Generally speaking, pull-requests are initiatives of individuals who want to provide solutions that might be useful for the project by contributing their particular knowledge of how certain issues can be resolved. Contributers need to demonstrate the value of their pull-requests before they are implemented. They can therefore be accompanied by large discussions via mailing-lists or other communication tools and sometimes trigger conflicts within the community.

If others determine the submitted modifications to be valuable to the project, the changes can be merged into the 'main-branch,' which is authoritative for big releases. Merging two branches requires a thorough understanding of the specific features of every branch and eventually leads to an expansion of the initial project. However, if

[6] For a detailed and visualized description of version control systems (in this case: git) and their capabilities, see https://nvie.com/posts/a-successful-git-branching-model/.

the maintainers of the master-branch reject the suggested modifications, it does not imply the end of this particular development branch. If the initial contributor (maybe together with a minority of other users) sticks to the assessment that their contribution is nevertheless valuable, they are free to continue to work on their own branch. In the long run, their modifications might even turn out to be more useful than at first assumed and will eventually be merged into the master-branch after all.[7]

Merging and branching are counterparts of cooperative development, incorporating high-frequent *just-do-it* as well as *trial-and-error* habits. They epitomize the differentiation and synthesizing of creative work that remains revocable and open for change.

All of this can be accomplished with decentralized version control systems. The technical functions I have outlined show how the idea of free software has led to the development of platforms that help facilitating the joint efforts of collaborative software development by making the contributions organizable with respect to quality control while they remain highly transparent to the public. Version control systems are thus an essential part of how the legal framework of the GPL is implemented as a practice of collaborative freedom. Christopher Kelty, an anthropologist who has published a book on the habitual practices of FLOSS, sees in developments like these the specific sustainability of the *Free Software Movement*, which goes beyond its mere ideological foundation: "The ideas of sharing and of common property and its relation to freedom must always be produced through specific practices of sharing, before being defended."[8] Kelty calls these complex interactions, arranged by a specific infrastructural framework, a 'recursive public': "Two things make recursive publics distinct: the ability to include the practice of creating this infrastructure as part of the activity of being public or contesting control; and the ability to 'recurse' through the layers of that infrastructure, maintaining its publicness at each level without making it into an unchanging, static, unmodifiable thing."[9]

For a concept of freedom that is based on the idea of sharing and on cooperative networks such as developer communities, the transparent and revisable development of

[7] If the differences of a branch to its initial master branch become significant, such branches sometimes happen to create a new and independent forked project, which – by the terms of the free license of the initial project – is required to keep its freedoms. It is thus technically and legally possible that the two projects later still share patches with each other. Otherwise, one is free to establish this branch as an own fork of the project, hoping to attract other developers. There are numerous well known software packages that have emerged out of forking processes like these. For example, the most popular free office suite *LibreOffice* is a second level fork, as it descends from OpenOffice.org which itself is a fork of StarOffice, a software suite which was popular in the 90s.

[8] Kelty (2008), 180.

[9] Kelty (2008), 62.

such platforms establishes the structural base of its practices. Without it, its idea of freedom would be limited to pure potentiality or completely miss an awareness for the requirements of concrete actions of freedom.

3. Requirements for Participation and Knowledge Communities

So far, I have outlined the emerging structures of collaborative development. These structures manifest the freedom to study and manipulate software code in a communal way. I have shown that the 'freedom' of *free software* isn't only rooted in the ideological and legal foundations of free licences but also in structurally maintained practices that depend on a critical mass of interaction.

But this interaction does not only happen between a few skilled programmers. The success of FLOSS is based on the fact that it has managed to implement ways for less technically skilled people to participate, for example by translating, sending in bug reports, and responding to user surveys. Even the mere usage of *free software* has driven the standardization of the internet's foundational communication protocols (TCP/IP) and data types.[10] In other words, both the usage of a given program as well as different forms of its co-development foster the dimensions of freedom that *Free Software* envisions.

In the following, I want to concentrate on the requirements of *active* and (co)creative engagement in FLOSS projects because its strategies of lowering the thresholds of engagement are insightful for a theological adaptation. In FLOSS, these thresholds are mainly localized on the level of abilities: Co-creative participation in the freedom of FLOSS is a question of knowledge, experience, and skills; that is, one needs to know how computer programs are developed, how code is written and how complex projects are designed.[11] The learning and teaching of these abilities requires the mutual sharing of knowledge, not only about a given project but also about how to connect and interact with its development community. The simple idea of freedom as a desire for transparency and openness is worthless if people are unable to benefit from it.[12]

[10] Kelty (2008), 166–7.

[11] Until a few years ago, even making use of the first freedom stated in the GPL (the freedom to use the software for any purpose) has only been practicable for enthusiasts who were eager enough to find out how to install and configure certain programs. In fact, until the mid-2000s the *Free Software Movement* was more concerned about security, functionality, and (as a recursive public) the political and social significance of its notion of freedom than about the implementation of user-friendly interfaces.

[12] Note, for example, the huge national disparity of pull-requests on one of the major version control systems *GitHub*: https://medium.com/@hoffa/github-top-countries-201608-13f642493773.

FLOSS culture realized this from its earliest days and understood that, in addition to its transparency that makes it a recursive public, it must face the challenge of enabling and empowering others. In other words, the mere sharing of code is not enough to produce a liberating effect from the idea of *free software*.

This is why the Free Software Movement has incorporated ways of mutual education since its appearance, evidenced in (sometimes excessive) documentations, highly frequented online forums, and various mailing-lists, all of which often provide a welcoming and supportive environment while fostering the quality of its contributions.[13] *Freedom of software correlates with sharing knowledge.* It is thus not coincidental that the idea of *Free Software* has influenced other knowledge-based sectors. The invention of a collaborative content management system for documentations, for instance, has set the foundation of today's most used encyclopedia: The technical infrastructure of Wikipedia is a derivative of what participants of *Free Software* already used decades ago for documentation purposes and it remains *Free Software* even today.

The implementation of data-mediated freedom through openness, transparency, and participation has not only transformed the way software is produced but also lead to the creation, evaluation, and spread of knowledge. The rise of Open Data, Open Science, and Citizen Science are prosperous examples of the entanglement of qualitative collaborative work with movements focused on education and knowledge.[14]

However, despite all efforts, it is evident that most implementations of freedom in FLOSS – especially the freedom to manipulate software and become creative in its development – are only performed by a few. Apart from a few enthusiasts, engaging in free software development remains an activity of professionals who are either paid directly to write code or need it for other professional tasks.

But the long-term effect of what FLOSS envisions is enormous, changing the ways people communicate and co-author the narratives of the digital. By inventing its own infrastructural basis of joint efforts, the concept of freedom within free software has affected the life of everyone who uses an online device. This shows how practices of freedom and their effects can be masked anonymously. But it also demonstrates that actively and explicitly offering *freedom in and of itself* might not be convincing to ev-

[13] There are numerous step-by-step-introductions for beginners and some platforms even provide lists of especially beginner-friendly projects. Cf. https://opensource.com/life/16/1/6-beginner-open-source.

[14] This goes beyond the ideological affinity of FLOSS and Open/Citizen Science, as the former often provides an appropriate or at least highly adjustable toolkit for large scale research. Cf. https://www.fastcompany.com/40569993/how-citizen-science-and-open-source-tech-can-create-change. See also https://opensource.com/article/18/5/citizen-scientists.

eryone in the same way – it all depends on how one can make use of it. The prevalence of free software shows that practices of freedom are required to reveal their immediate practical use, to provide reasons why someone should spend the time, energy, and creativity to leave the seemingly safe haven of proprietary software,[15] which actually restricts not only developers but also the users in a way that they are often not even aware of.[16]

4. The Risk of Competing Visions

The idea of free software developed as a reaction to limited resources, working hours, and technological knowledge, and through the creativity of individuals eager to chase after big visions of technological development. By releasing software under free licenses, people like Linus Torvalds, the inventor of the Linux kernel, opened up development processes to the public. They trusted the positive effects of crowd-based co-creation driven by the commitment of individuals who share their experience and knowledge.

But licensing software under a public domain must not be misunderstood as the simple distribution of programming tasks to an arbitrary public. Although this might be one of the initiator's interests (especially when FLOSS practices are adopted by commercial software companies), the consequences of public licences are much more unpredictable. To dispense with copyright is to dispense with one's exclusive decision-making authority. It implies a switch from a model of ownership to an open process of co-authorship with unforeseen outcomes.

On an individual level, releasing software as *Free Software* means to take the risk that the very work one values as useful and powerful enough to be published *will* be criticized, adapted, or even misused by others. Although the main currency of FLOSS practices is public recognition and reputation, the publication of code snippets requires the admission that the results are open. As described before, this openness to competing imaginations and visions can ultimately lead to division within communities (*forking*) and is a frequent cause of personal frustration. This can occur due to a lack of response to pull requests into which people have put energy, due to a lack

[15] This struggle has been examined in a qualitative study in the field of creative graphics design, cf. Velkova (2016).

[16] This is the reason why the *Free Software Foundation* has launched a rather polemical campaign that raises awareness about the several dimensions of restrictions that, for example, Microsoft puts on their users; cf. https://www.gnu.org/proprietary/malware-microsoft.html.en.

of understanding concerning the demands of a project, or simply because of political issues.[17]

But FLOSS practices don't only reveal individual vulnerabilities. FLOSS practices have been widely adopted by a lot of software companies, which expect positive effects from encouraging the public to participate in the development of their products.[18]

Of course, this poses a potential risk to the driving ideological ideas behind *Free Software*. Its notion of freedom is mediated by software and therefore mainly independent from its engaging subjects. But this makes it highly vulnerable to shifts of power, for instance when whole companies enter the field with a decisive business plan that becomes authoritative.[19] Although a free share-alike-license (which demands that further developments have to stay licenced as free) legally guarantees that a FLOSS project cannot be turned into proprietary software, the funding and organizational leadership of big players still has a strong influence on the dynamics of the project.[20] On a small scale, this can influence practices of writing and implementing code; in the long run, specific ideas of a certain company or patron influence the whole project. This certainly corrupts the idea of equally competing visions and the openness of the development process, the most persuasive element of FLOSS.[21]

That is exactly why the conflict between *Open Source* and *Free Software* plays a significant role for the question of how freedom can be sustained. While the *Open Source Initiative* attaches more significance to the actual practices of collaborative development, the *Free Software Movement* is additionally concerned with the explicit conception of these practices as *practices of freedom*. In theological terms, we could call the *Free Software Foundation's* implementing and sustaining of practices of freedom its *doxology of freedom*. Because of it, the *Free Software Movement* places such great emphasis on the label 'free,' which communciates its ideological background and its

[17] https://github.com/stereobooster/react-snap/issues/103

[18] One of the significant milestones in FLOSS was undoubted the release of the code of the Internet Browser *Netscape Navigator 4*: https://www.oreilly.com/openbook/opensources/book/netrev.html. Other examples are the development of the Online Learning Platform *Moodle* (http://oss-watch.ac.uk/resources/cs-moodle) or several software products of the Open-Source-Company *Red Hat*, including an enterprise Linux distribution, which made it one of the most profitable FLOSS driven companies: https://www.wired.com/2012/03/red-hat/.

[19] Lately Amazon has distinguished itself by causing a lot of frustration among the open-source-community: https://www.businessinsider.com/amazon-responded-to-a-frustrated-open-source-developer-2019-6?IR=T. This demonstrates the dangers of FLOSS-practices that neglect the incorporation of strategies that secure these practices and protect their actors from being exploited.

[20] For an example of a share-alike licence, see https://creativecommons.org/licenses/by-sa/4.0/.

[21] Cf. https://techcrunch.com/2018/11/29/the-crusade-against-open-source-abuse/.

visionary narrative.[22] Sustainability is not a mere wish; the proclaimed vision of a just society is actively carried out through free and collaborative knowledge production. In contrast, *Open Source* is generally more interested in the direct effects of dealing with the openness of the source code, without missionarily supporting the ideological basis of redeeming societies from proprietarily distributed information. In other words, while *Open Source* is mainly interested in spreading the concrete practices of FLOSS in order to foster high quality software through the experience and skills of the crowd, *Free Software* shows a tendency to spread its idea of freedom by directly and deliberately countering proprietary modes of development. It highlights the intertwining of practices of freedom with its praise. It thus does not simply trust in the system-immanent powers of self-spreading freedom, but it openly and directly faces the challenges of commercial occupation and the assimilation of its practices through other ideologies. It incorporates constant and open competitions of different visions through its doxology of freedom.

5. Theological Resemblances

Free Software is neither a nominalistic claim nor a mere collection of habits of interaction. Rather, as I have shown, *Free Software/FLOSS* has been able to derive concrete practices and sustain concrete structures from a distinct concept of freedom. This observation marks the initial point of my examination of analogies between *Free Software* and Protestant theology.

Ever since the Reformation, Protestant theology has referred to certain notions of freedom in order to describe the Christian faith as well as its dogmatic, ethical, and existential implications. In the following sections, I want to examine resemblances between the above described structures of intertwining freedom claims and the shaping of practices of theological freedom within Protestantism. For that purpose, the analysis of FLOSS culture serves as a spotlight for the texture of Protestantism and its embedded practices of freedom. It makes visible certain freedom practices in Protestant theology that resonate with FLOSS and illuminates their respective differences. What follows is an endeavor to search for analogies and contrasts between FLOSS and Protestant theology.

My considerations are based on the observation that practices of freedom in FLOSS are concrete communal (inter)actions. This resembles the Pauline understanding of

[22] Cf. Stallmann, Why Open Source misses the point of Free Software: https://www.gnu.org/philosophy/open-source-misses-the-point.html.en.

faith in Christ, which is fostered in communities of faith in the presence of the Holy Spirit (Rom 8:5–11, 1. Cor 12:12–30). This faith can only be understood in concrete communal (inter)action. On several occasions, Paul highlights the significance of freedom in Christ, a freedom that inevitably leads to the formation of communities where people come together to serve another with their gifts and virtues. In Galatians, for instance, he refers to the freedom from the obligations of the law and from the desires of the flesh, leading to the fruits of the spirit, which characterize the spirit of the community (Gal 5:13–25). Thus, I am not following the subjectivist idea of negative freedom (as mere independence) in favour of an approach that values openness and co-creativity as characteristic for practises of freedom through faith. Consequently, freedom is understood in its pneumatological and ecclesiological contexts: A biblically oriented theology of freedom is about the implementation of practices of freedom that tend to shape communal existences. This existence is characterized both by an openness to its further development by and for its participants and by the building of structures that foster this very freedom in a communal way.

5.1. The Struggle of the Reformation against Proprietary Distributions of Orthodoxy

In a first step, I want to analyze specific adjustments of the Reformation as the implementation of practises of freedom, practices that free the promise of salvation by faith from its proprietary distribution. On this first level, we can observe similar structures in both contexts: FLOSS and Protestantism will appear to be analogues.

It was the Reformers' struggle to challenge the copyright of Christian orthodoxy in order to rectify the heretical practice of indulgence trade. The foundational modification performed by the theologians of the Reformation was only possible by implementing a practice of theological freedom that denied the exclusive authority of religious and theological authorship of the Roman Catholic Church.[23] This directly resembles *Free Software's* paradigm of decentralization in decision-making by denying any sort of copyright and releasing software code into public domains.

In his treatise *On the Freedom of a Christian*, Martin Luther deals with the same issue by questioning the centralized restrictions of the Roman Church of his time from an

[23] In his study on the sociological structures of FLOSS, Christopher Kelty points out that even the free-software-movement itself refers to the parallels of the Reformation as a common narrative, for example when *geeks* identify their own struggle against proprietary software with the reformers st against the Roman Catholic Church. Cf. Kelsy (2008) 64ff.

anthropological and christological perspective. His dialectical argument opposes a soteriology in which salvation is externally restricted and regulated – historically by the religious demands of the Roman Church. In a first step, Luther's writings on freedom therefore establish the negative freedom of a Christian whose faith frees them from a soteriological point of view. This approach is then constructively developed in two theologoumena, which lead to a positive understanding of freedom and render Protestantism's vision for theological authorship. It is the combination of the mandatory scriptural principle and the non-restrictive priesthood of all believers that transform the Reformers's theology of freedom into a mode of doing theology.

The scriptural principle as an epistemological proposition initially leads to what Matthias Gockel has called "a theology of open sources."[24] It constitutes the referential standard for all theological search for truth and dogmatic authorship. Luther's emphasis on the importance of the linguistic methods of his time shows that theological authorship on the basis of Scripture must be implemented in a controlled, comprehensible, and therefore transparent way.[25]

The scriptural principle was accompanied by a christologically grounded understanding of priesthood, the second foundational implementation of theological practices of freedom. With reference to 1 Peter 2:9, Martin Luther identified Jesus Christ as the one and only priest. This means that human beings can be called priests only through their participation in Christ.[26] This is the root of the theologoumenon of the priesthood of all believers, a concept that creates a general field of tension between the exclu-

[24] Cf. Gockel (2018).

[25] Also historical exegeses itself can be understood as an examination and reflection on cooperative freedom practises of the biblical authors and editors. Through the eyes of FLOSS, the historical development of the biblical texts can be understood as the manifold extension and adaptation of testimonies. The different layers of editorial work proves the existence of this very freedom practice among biblical authors and editors, to engage with previous traditions and thus showing their testimonial and theological relevance by taking their specific circumstances into account. The freedom of the biblical canon even tolerates the existence of several branches: While covenant code, deuteronomic code, and holiness code present different development branches of the law in the Hebrew Bible, the four gospels can be seen as an equivalent in the New Testament. According to the two-source hypothesis Mt and Lk can roughly be seen as different merge results of Mk and Q. Thus, historical research on the biblical text does not just satisfy the mere curiosity about the history of some ancient texts. It also undertakes the task of unfolding the development process of the binding testimonies of first grade. Thus, the work of historical biblical studies is comparable to the solution of version control systems: both of them reveal the complex dependencies and motives of multiple actualisations and adaptations that include external material into the new stage. It shows how the development of the very early texts of Christianity has been characterized by co-authorship of the biblical editors and their engagement in a canonical conversation. "This doesn't mean that it is all opinion, but recipes, like the biblical narrative, required a number of hands and voices to alter and arrange it prior to its current form" (Ott [2014], 144).

[26] "Wie nun Christus die Erstgeburt innehat mit ihrer Ehre und Würde, ebenso teilt er sie allen seinen Christen mit, dass sie durch den Glauben auch alle Könige und Priester mit Christus sind." Luther (2012), 295.

sive singularity of priesthood in Christ and its universalisation in all who are baptized. It marks the area in which concrete practices of freedom may be localized in an ecclesiological and pneumatological manner.

While Luther and Calvin advocate for a functionally structured church through the provision of ministry, Ulrich Zwingli, the Zurich Reformer, explicitly offers a distinct pneumatological approach. He takes up Luther's concept of the freedom of a Christian and emphasizes the significance of the work of the Spirit, which enables human beings to read the Bible as the Word of God without the guidance of the Church – or even against it, if it misses to perform its duties. He therefore identifies the involvement of the non-ordained as a liberation from the moral and clerical restrictions of the Roman Church:

> This will help all those who adhere to the Holy Scripture, who stand up to the enemies of the Scripture. So read and understand, open the eyes and ears of the heart! Listen and see what God's Spirit is saying.[27]

For Zwingli, the freedom of a Christian therefore establishes human practices of engaging with the Bible, practices that he interprets pneumatologically. Moreover, Zwingli identifies the work of the Spirit within these very processes of religious and theological learning through reading Scripture. This notion later became known as the *testimonium spiritus sancti internum* (the internal testimony of the Holy Spirit).

He supports this with the confidence that an appropriation of biblical texts is not an arbitrary but a Spirit-led update from which the individual's understanding of the Word of God derives.

Such a pneumatological interpretation of the theologumena of the scriptural principle and the priesthood of all believers shows parallels to the first two freedom claims of the free-software license, which establlish the freedom to use and to study a given program. Analogous to Zwingli, the liberating work of the Spirit empowers individuals to acquire, study, and interpret the biblical texts. However, in order to turn this empowerment into a freedom practice of the masses, enormous challenges in terms of accessibility have to be faced. Thus, it was only consistent that the Reformation went hand in hand with translations of the Bible, the development of the letterpress, and the encouragement of ordinary people to learn and to read.

[27] Zwingly (1995/1522), 22.

However, a mere individual interpretation and application of the biblical texts can only be understood as a first phase of a theology of freedom. After all, Protestantism is characterized not only by the individualization and particularization of religious and theological continuation. The next step, therefore, is to ask about ecclesiological practices of freedom in light of the analysis of FLOSS culture.

5.2. Software/Ecclesia Semper Reformanda

While the history of FLOSS shows how freedom enables individuals to study program code and perform adaptations for their personal needs, it is also engaged in the formation of institutionalized platforms that shape the understanding of the freedom of software. Public version control systems, for instance, are concrete implementations of freedom practices that enable people to engage with each other's impulses for improvement and development. They are the consistent embodiment of the fact that free software is *software semper reformanda*. It can only draw on the full potential of its free(ing) license by fostering the creative and competent engagement of a multiplicity of contributors. It relies on adequate environments and an infrastructure that brings those contributions together. To meet this need, FLOSS has created recursive publics, the basis of collaborative evolution, which are able to handle the concrete adjustments in the code by executing pull-requests. We might look for analogous processes within Protestant theology by asking what structural implementations of the freedom of a Christian it has developed to perform the idea of *ecclesia semper reformanda*.

Protestant traditions offer multiple models for how this theologoumenon can be implemented theologically in the social structure of the church. One of them is Friedrich Schleiermacher's ecclesiology, which highlights the importance of the mutual sharing of religious experience within the community of the church. Schleiermacher argues that only the rich plurality of individual impressions can approximate the redemptive work of Jesus Christ.[28] He also claims that the shared (and therefore supra-individual) religious consciousness of the community is the Holy Spirit itself.

Despite the potential of a fundamentally egalitarian approach to biblical hermeneutics through pneumatological interpretation, most Protestant thinkers have seen the need to organize the complexity of the church, establishing structures that secure its visible persistence. A challange that returns whenever the church has to conquer

[28] Cf. Schleiermacher (2008/1831), 299.

heretical and harmful influences that would corrupt its nature as a community that derives its communal spirit from its freedom in Christ.[29]

Within this tension between securing structures and a non-restrictive approach to hermeneutics, it is insightful to look at those ecclesiological approaches that have sought to implement practices of freedom precisley through the institutionalized structures of the church.

One prominent example is the German Lutheran theologian Wolfgang Huber, whose institution-theoretical approach claims that freedom within Christian theology should be understood within a communal paradigm: "It is realized in community and in mutual understanding, in *communio* and *communicatio*; thus, it may be called 'communicative freedom.' "[30] That is to say, Huber locates freedom within the concrete shapes and actions of communities that individuals engage in. For Huber, this applies to all sorts of communication within the church, may it be religious, moral, or theological.

The problem with this concept is that there is a lack of concrete implementations of structures that actually promote this communicative freedom and its further development. Huber's ecclesiology ("church of freedom") focuses on *installing* structures of freedom, but it does not develop an adequate concept for ensuring their continuing developmental openness. Although Huber does mentions the tool of language, he does not pay enough attention to the dynamics of power *within* the empirical church.[31]

In this context, FLOSS culture can serve as a contrasting template that shows why Huber's ecclesiology lacks a proper implementation of practices of freedom. We have seen that practiced freedom is always linked to enabling structures through appropriate platforms (recursive publics). Their important task is to implement circular movements of irritation and external impulses by providing interfaces for individuals to contriute their visions and suggestions for improvement. The most successful of such platforms have emerged from concrete needs and a knowledge of the communicative

[29] On the Reformers' struggle to maintain the teaching of freedom against Rome's doctrine, cf. Calvin, Offices of the church and their pneumatological foundation (Calvin, Inst. IV,3,2). Another example is the German Church Struggle, which shows the difficulties of a church that is endangered to submit to National Socialist ideology. Cf. Barmen Declaration VI.

[30] "Sie verwirklicht sich also in Gemeinschaft und in wechselseitiger Verständigung, in communio und communicatio; deshalb kann sie 'kommunikative Freiheit' genannt werden" (Huber [1983], 118).

[31] This was one of the decisive critiques of the impulse paper "Church of Freedom" by the EKD, which was presented by Huber as its former president; see Kirchenamt der Evangelischen Kirche in Deutschland (2007).

specifics. The success of FLOSS is based on a bottom-up development principle that relies on the particular, non-restrictive involvement of additional contributors.

An ecclesiology of freedom that seeks to learn from the successful cooperative practices of FLOSS may therefore point to a systematic appreciation of co-creative dynamics and to emerging structures maintained by the participants themselves. In computing as well as in Christian communities, this implies the necessity of educational processes that cultivate and perpetuate an expressiveness that leads to the emergence of recursive publics, which in turn enable individuals to hand in high quality religious and theological pull requests.

Churches of freedom need grassroot structures that allow for broad religious and theological literacy at an eye level without undermining the different parameters of the various contributions. For it is the variety and speciality of contributions that drive the quality of both technical and theological knowledge production.

On a parochial level, this could be exemplified by an appreciation for communicative, decentralized forms of community. Movements like *Fresh Expressions* and *Emerging Church* have developed reasonable alternatives to the model of the 'people's church' (*Volkskirche*), which is driven by the vision for an all-compatible program. These movements try to establish platforms of theological co-authorship through flat teaching hierarchies, the sharing of life experiences from various contexts, and the sensible evolution of religious practices.[32]

On the specific level of academic theology, approaches such as *Citizen Theology* pursue the vision of implementing pull-requests that integrate the diversity of Christian forms of life and religious knowledge in a multidirectional way.[33] Context-based learning from theological adaptations to specific requirements puts one in an epistemologically favorable position and thus function as a starting point for theological pull-requests. This vision of a systematic implementation of pull-requests is driven by the pneumatological assumption that the teaching of the Spirit does not only act within singularities but through engagement – by sharing religious and theological knowl-

[32] It needs to be mentioned that the biggest difference between FLOSS and any religious cultures such as Protestantism is the aspect of timing, speed, and frequency of change. Software development fosters rapid development and sometimes even forces its users to adapt to recent changes. Changes can even be made on a trial-and-error-basis that easily risk to lead into a dead end if they turn out as insufficient. Contrary to this, transformations within religious communities need to take long grown and tenderly fostered traditions that people identify with into account. This makes changes in religious communities and their theological reflection slow, sometimes even too slow, for example when adaptations come to a halt and communities and their theological reflection are unable to keep up with their environment. Cf. Schleiermacher (1910), §§203.204.

[33] Cf. Friedrich, Reichel, and Renkert (2019).

edge beyond certified expertise. Rightly understood, theological authorship is always theological *co*-authorship, as reflective assessments about the Christian faith derive from shared experiences and contextual insights into the meaning of the biblical traditions.

But such an epistemic adjustment is not without risk. The open source movement exemplifies that the structural integration of cooperative practices does not necessarily have to support its ideological foundations. The ongoing dispute between *Free Software* and *Open Source* shows this quite well. This correlates with the question about the significance of both the orthodoxy and the doxology behind these freedom practices. This concern, however, is not a sufficient reason to completely abandon such models. It is a question that every model of 'church for the world' has to deal with.[34]

The development of a theology of freedom in the context of ecclesiology builds on the idea of religious and theological co-authorship. This idea, in turn, needs to be properly implemented in social structures of the church and in the methodology of its theologies. It understands these structures and methods as practices of freedom and – fully aware of the risks – relies on the promise of the Spirit's presence through the various charisms of the members of the Body of Christ. Protestantism needs this breadth of authors in order for its theologies to be enhanced, constructively challenged, and further developed.

5.3. Ecumenism of Branches and Merges

On a third level, I want to use FLOSS culture as a contrast medium to elucidate Protestantism's specific inability to secure its own epistemic standpoint. This makes embracing the Spirit's freedom a necessary consequence.

In order to do so and to visualize the scope of this section, I want to concentrate on the structure of how cooperative development takes place in FLOSS. As described above, decentralized development in FLOSS often involves the simultaneous execution of

[34] One might read the story of the healing of a bleeding woman (Mk 5:25–34 parr.) as the synoptic gospel's sensibility for the question of mere profiting from the beneficial effects of faith. Moreover, the narration of the dispute of Jesus with his disciples about the legitimacy of the alien healer can be seen as one possible account on this problem: Mark clearly states his non-restrictive approval of the healer through Jesus's answer: "Do not stop him, for no one who does a mighty work in my name will be able soon afterward to speak evil of me. For the one who is not against us is for us." (Mk 9:39–40). Contrary to Mark, Matthew alliterates Jesus's statement in a slightly different context when he explicitly demands an affiliation: "Whoever is not with me is against me, and whoever does not gather with me scatters" (Mt 12:30).

different developmental steps by a variety of people. Small-scale changes are being outsourced to branches and eventually will be merged back into the master branch.

By comparing the practice of this decentralized process of *branching* different directions of development with the generation and growth of theological traditions, we can unveil a specific blind spot of theology, namely its non-foundationalism. To understand the contrast to FLOSS, we first need to identify the *tertium comparationis*, which lies in the analogous freedom practice of the separate, yet parallel development of different branches. What version control systems make possible for collaborative code manipulation can also be seen in the history of ecumenism: Theology, not only understood as an analytic but a constructive enterprise, is a vital continuation of differentiated yet interdependent co-authorship. Different theological approaches or even denominations can be envisioned as branches that continue theology not only as a linear development but as independent, parallel, and alternative developments of theology.

Examples for externally driven developments are theologies that point to contextual issues: The emergence of liberal traditions in the nineteenth century were a response to the Enlightenment philosophy in Western Europe. Likewise, the innovations of various liberation theologies have emerged from certain life contexts and experiences of poverty and oppression. Internal reasons, by contrast, are the systematic detection of theological blind spots within one's own dogmatics or also new exegetical insights that attempt to rectify certain theologumena. Of course, in most cases external and internal reasons concur. The history of the diversity of theological and religious traditions of Christianity can be read as a complex network of different branches that are sometimes loose- and sometimes close-knit.

This last point shows that the theological tradition isn't only one of mere differentiation (branches) but also one of mutual interdependence and stimulation through theological difference (merges). Theological encounters of different branches promise the possibility of mutual correction. The conversation between different approaches and traditions in the search for theological knowledge may turn out to be quite conflicted or even disruptive, and merge attempts usually pose some big challenges for the involved branches. What, in software development, signifies a time-consuming process of merging-conflicts, is, for the *fides quaerens intellectum*, the place of a constant and not always consensual search for truth.[35] Not only the church but also its reflective enterprise *theology* is an endeavor *semper reformanda*.[36]

However, particular merge processes are not only found in the context of explicit theologies. Occasional merges also occur in implicit theologies, in the practical formation of ecumenical or inter-religious encounters. And, realistically speaking, this often does not result in a success story of common consensus. Another closer look at the experiences of FLOSS can illustrate this. Large projects like the Linux kernel, for example, have a massive number of branches with dead-ends. Due to their technical, stylistic, or political inadequacy they are never merged into those critical branches that attract the interest of the public. Of course, this raises the question of power – for both software production and ecumenism alike. The maintenance of merges, just like the encounter between different theological developments, does not happen in an egalitarian way. Like it is possible that the decision-making in the 'master branch' of a software project is undertaken by a company or a patron, we can observe similar tendencies in the writing of theology.

But this is also where the analogy ends and where Protestant theology shows its decisive contrast in its practices of freedom: By renouncing a synthesizing and boundary marking organizational unity (after all, Protestantism knows of no central teaching position), it also lacks any empirical organizing reference.

Theology, rightly understood, simply does not have a tool that would allow it to locate its own branch relative to a master-branch. Christian faith does not operate within in the category of ownership, but only by means of authorship and co-authorship. Without any institutionalized and theologically legitimized teaching position, there is neither a distinct maintainer of the master-branch, nor is there anyone who could even identify any branch as the universal master-branch without manifesting a paradox of freedom practices. In contrast to the clearly localizable structure of the branches of a development tree in a software version control system, the Protestant epistemological principles lack the possibility of an independent verification of their own branches. I have demonstrated this *inability to verify* above by describing the inevitable tension of

[35] It should be noted that in software development merge-conflicts are not exclusively based on the simple principle "It is right if it works." Analogous to the dispute over methods and verification of theological statements ("what is good theology?"), software development has an open debate about 'refactoring,' its key question being: "What is good code?"

[36] Realistically, one has to admit that merge processes in theology are often not carried out with respect to an entire branch, but with a great deal of particularity. But targeted impulses from other theological and religious traditions can redirect theological thinking's attentiveness. As an example, see https://www.gnu.org/proprietary/malware-microsoft.html.en.

the scriptural principle and a christological reasoning of the priesthood of all believers. Post-theistic theologies, as well as theologies based on the openness, changeability, and liveliness of God, will renounce a verifiable reference to a master branch. And they will do so not only for epistemological but also for theological reasons.

The pneumatological assumption on this third level is that the self-unfolding presence of the Holy Spirit is not only to be located within the boundaries of what we call church (which, in particular, can mean one's own religious, denominational and contextual bounds), but that the presence of God acts within the transgressions of these epistemic borders. However, without assuming a blurred and indistinct presence of the Spirit, the work of the Spirit can be seen as a force transcending the boundaries of social and therefore epistemic self-affirmation. In addition to Schleiermacher's notion of the Holy Spirit *within* the communal spirit of the church, it is therefore adequate to also hope for the presence of God's Spirit in the differentiated intersections of mutual ecumenical learning. Practicing theological freedom requires taking the freedom of the Holy Spirit into account as well as the fact that this freedom might unfold within unknown contexts that themselves testify to the Spirit of faith, love, and hope. These testimonies of *others* might eventually turn out to be a more adequate description of, and even an impulse for solutions to, one's own theological and religious quests. This re-localization would lead from a pneumatology of the *spirit in nos* (as observed on the first and partially on the second level of the constructive part of this paper) to an understanding of the *spirit extra nos*. That is to say, toward an *inter nos* in the encounter of separate branches. While software production mostly follows the logic of technical compatibility and efficiency, theological development should not only be functionalized for its practical feasibility but also for the question of truth. This is why the question of the discernment of the spirits in light of faith, love, and hope becomes the crucial question, and it is deeply entangled with what I have described so far. The assumption of the *spirit extra nos* and the mere presence of *the other* is no guarantee for the enlightening and self-revealing work of the Spirit – and neither is a mere communal spirit of one's own branch. It is therefore necessary to understand theological freedom not only in the sense of independence but as a co-dependent engagement with the *source code* of the Christian faith rooted in the engagement with the pluriform and many voiced biblical canon.

A small remark about the eschatological implications of this: The idea of possible and particular merges must not be understood teleologically. The eschaton is not to be envisioned as a super-merge that re-includes every single branch. It is rather the justification of the diverse and multiple endeavors to conceive the reality of God within this world through the contextual exploration of the biblical promises.

6. Conclusion

In order to show how FLOSS culture can be compared with the fundamental theologoumena of Protestant theology, I have outlined the entanglement of a proper concept of freedom with the practices of freedom it entails. I have then described the concrete phenomena in FLOSS that have proven to be successful and influential for the ways software is nowadays developed.

Understanding theology as a creative enterprise, the category of authorship has turned out to be a better category than ownership in terms of its development. After comparing how software development and theological development, according to the fundamental convictions of the Reformation, implement structures of free and co-creative engagement, I described three levels of theological practices of freedom that resemble the fundamental insight of FLOSS, which is the conceptual connection between a general statement of freedom(s) and its implementation in concrete practices.

The first level is located in the combination of the scriptural principle and the priesthood of all believers. It is the practice of a fundamentally non-restrictive openness of the Bible that enables individuals to engage with the foundational texts of the Christian faith.

On a second level, I have shown how the social structures of the church and the development of its theology can resemble the idea of the 'Freedom of a Christian.' In order to foster the freedom of software, FLOSS has developed recursive publics. In this, it can serve as an example for Protestant ecclesiology, calling it foster structures that embody the communal aspects of theological co-authorship.

In the last section, I have compared the practices of *version control systems* (branching/merging) with the mutual influence and interference of different theological developments within ecumenism. In consequence, the epistemological uncertainty of one's own branch in relation to others has turned out to be one of the major differences between FLOSS and Protestant theology. Gaining theological knowledge therefore depends on the constructive transgressions of denominational and cultural boundaries.

These different levels on which we can identify theological practices of freedom are interdependent and may be understood in a pneumatological way: By professing the

Freedom of a Christian and developing its corresponding practices of freedom, Protestant theology expresses its faith in the plural and differentiated presence of the revealing Spirit. Theological enterprise is not driven by the aim of mere innovation, after all, but is quest for knowledge. A quest that can only be carried out by a free development of theology cultivated in cooperative practices of freedom aimed at grasping the plurality of the self-revealing presence of the Spirit.

Bibliography

Friedrich, Benedikt, and Reichel, Hanna, and Thomas Renkert. 2019. "Citizen Theology: Eine Exploration zwischen Digitalisierung undtheologischer Epistemologie." In *Digitaler Strukturwandel der Öffentlichkeit. Interdisziplinäre Perspektiven auf politische Partizipation im Wandel*, edited by Jonas Bedford-Strohm, Florian Höhne, and Julian Zeyher-Quattlender, 175–192. Baden-Baden: Nomos.

Gockel, Matthias. 2018. "Ad fontes: Zu einer Theologie der offenen Quellen." *Cursor_ Zeitschrift Für Explorative Theologie 1*. https://doi.org/10.21428/d1d24432.

Huber, Wolfgang. 1983. "Freiheit und Institution: Sozialethik als Ethik kommunikativer Freiheit." In *Folgen christlicher Freiheit*, edited by Wolfgang Huber, 113–127. Neukirchen-Vluyn: Neukirchener.

Jüngel, Eberhard. 1978. *Zur Freiheit eines Christenmenschen: Eine Erinnerung an Luthers Schrift*. München: Kaiser.

Kelty, Christopher M. 2008. *Two Bits: The Cultural Significance of Software*, Durham: Duke University Press.

Kirchenamt der Evangelischen Kirche in Deutschland (EKD). 2007. *Kirche der Freiheit: Perspektiven für die evangelische Kirche im 21. Jahrhundert: Ein Impulspapier des Rates der EKD.* Hannover.

Luther, Martin 2012. *Von der Freiheit eines Christenmenschen (1520).* in *Martin Luther Deutsch-deutsche Studienausgabe: Band 1. Glauben und Leben* edited by Dietrich Korsch. Leipzig: Evangelische Verlagsanstalt.

Ott, Kate M. 2014. "Creating 'Open Source' Community: Just Hospitality or Cyberspace Ivory Tower?" In *21st Century Feminism: Technology and Community*, edited by Gina Messina Dysert, London: Routledge.

Schleiermacher, Friedrich Daniel Ernst. 2008. *Der christliche Glaube nach den Grundsätzen der evangelischen Kirche im Zusammenhange dargestellt.* Zweiter Band (1831). Berlin: de Gruyter.

Schleiermacher, Friedrich Daniel Ernst. 1910. *Kurze Darstellung des Theologischen Studiums zum Behuf einleitender Vorlesungen* (1830), edited by Carl Stange. Leipzig: Deichert.

Stallmann, Richard. *Why Open Source Misses the Point of Free Software.* ttps://www.gnu.org/philosophy/open-source-misses-the-point.html.en (accessed Jan 23, 2022).

Velkova, Julia. 2016. "Free Software Beyond Radical Politics: Negotiations of Creative and Craft Autonomy in Digital Visual Media Production". *Media and Communication,* 4(4): 43-52, https://doi.org/10.17645/mac.v4i4.693.

Welker, Michael. 1992. *Gottes Geist.* Neukirchen-Vluyn: Neukirchener.

Zwingly, Hyldrich. 1995. "Die freie Wahl der Speisen" (1522). In *Schriften I* edited by Thomas Brunnschweiler and Samuel Lutz, Zürich: TVZ.