WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

Seminar paper at the Department of Information Systems and Operations Management

Proprietary vs. Open-source Software Markets in IT: Apple, Microsoft, Google

Gregor Koppensteiner

Wien, 17.12.2020

EFMD
EQUIS
ACCREDITED

Copyright clause

Please read carefully and sign before submitting the work!

I certify by my signature,

- that I have written this thesis independently, have not used any sources or aids other than those indicated, and have not made use of any other unauthorized assistance.

- that the present work has not been submitted for review in any form as an examination paper to an assessor in this country or abroad.

I am aware that any violation will have consequences under both study law and criminal law. Immediately, a copyright violation will result in a negative exclusion from the course in question as well as an immediate report to the Office of the Dean of Studies of the Vienna University of Economics and Business Administration and to the authors affected by the plagiarism.

Signiert von: Gregor Koppensteiner

Datum: 17.12.2020 10:50:36

Dieses mit einer qualifizierten elektronischen Signatur versehene Dokument hat gemäß Art. 25 Abs. 2 der Verordnung (EU) Nr 910/2014 vom 23. Juli 2014 ("eIDAS-VO") die gleiche Rechtswirkung wie ein handschriftlich unterschriebenes Dokument.

Dieses Dokument ist digital signiert!

Prüfinformation:
Informationen zur Prüfung der elektronischen Signatur finden Sie unter:
www.handy-signatur.at

www.a-trust.at

TRUST einfach sicher

Gregor Koppensteiner
(h11801956)

Table of contents

List of tables

List of abbreviations

OSS	open-source software
OS	open source
OS (in chapter 6)	operation system
PS	proprietary software
TCP	total cost of production
TCO	total cost of ownership

1 Introduction

The innovation processes applied in the software field are already widely discussed. The successful phenomenon of open-source software (OSS) raises new research questions about whether and how the free circulation of ideas advocated by the movement and the collective handling of intellectual property rights perpetuate innovation. The ideological counterpart to this tendency is the use of proprietary software, which is positioned at the other end of the spectrum.

The purpose of this paper is to provide a foundation to build these research questions on. This seminar paper discusses these two development approaches in terms of their principles, their historical background, and how they are used and disseminated in information technology. In terms of practical use, the three major IT companies, Apple, Microsoft and Google, are examined to identify their position on this topic. This investigation requires an economic reference. Therefore, the proprietary and open-source development approach are put into an economic context - from the perspective of the vendor and that of the customer.

The studies published on this topic so far are roughly divided into two categories: a product-related and an economy-related interpretation. Another objective of this work is to combine these two approaches, as they are not necessarily excluding each other.

The underlying research question of this seminar paper is: What is the specific nature of proprietary and open-source software and how do they coexist in an economic context?

2 Theoretical principles

The past forty years can be characterized by increasingly rapid advances in the field of providing software. Over time, two concepts have become established that could not be more contrary: proprietary and open-source software. In the following chapter these two theses are first outlined and examined according to their principles. Thereby, attention is also paid to the historical background.

The research was mainly based on journal articles, selected book chapters, and academically objective websites that have provided valuable insights over the past 25 years to answer the stated research question.

2.1 Proprietary Software

There is a relatively small amount of literature dealing with relevant proprietary software. The general understanding that the "manufacturing method" of a product lies with the producer is also assumed for immaterial goods. And this is exactly what proprietary software, also called closed-source software, is meant by.

In this context "manufacturing" refers to the source code and its accessibility and modification. Proprietary software is software which is owned by the developers. It is therefore subject to copyright laws, and only the author or owner has control over its development, just like with any other product. Producing proprietary software provides a clear business model – the owners sell their product and make money with it (Thompson, 2020).

The actual product is the source code, which is seen as a competitive advantage and at the same time a trade secret. To protect this asset, the user is deprived of the freedom to redistribute, examine and execute the software for any purpose. From the developer's point of view this is ensured by the copy prohibition via different contractual regulations, such as EULA, the End User License Agreement (Webcampus.de, 2017).

Closed-source software, as already mentioned, is characterized by a number of restrictions for the user. These can only be executed by the distributor. Therefore, the ownership rights must be secured from each programmer who may have co-authorship rights in the software. This is because copyright ownership automatically vests in the individuals who create the work, unless the work is a "work made for hire." Therefore, it is common practice in software companies for independent contractors who are commissioned to develop software to obtain a written assignment of the contractor's rights to the software to ensure that it is actually transferred to the company (Finkel, 2017).

2.1.1 Historical Background and Principles

The way software understands itself has changed continuously since the beginning of computerization in the 1960s.

The first computers at that time would hardly be recognizable in terms of hardware and software as their pendants of today. They were huge machines that took up entire rooms, which had to be specially cooled. The computers were primarily used to process high-volume data, and because they were so costly, they were often leased to corporate clients rather than being sold. All software installed on the computers was also supplied by the vendors, who also provided the source code.

Even though there was only a handful of customers at that time, it was nevertheless possible to program parts of the system on one's own. This procedure became the norm at that time, as it made it accessible to interested parties (especially research institutions and universities) to participate in further development.

However, this very collaborative approach has been legally restricted by declaring computer programs as intellectual property of their authors. At this point software gained the same status as literary works, subjecting them to the same copyright laws.

This status cleared the way for the closed-source software business model and marked the beginning of software licensing. Software moved away from the collaborative development model, and in the late 1970s and early 1980s it became standard practice to charge for software licenses.

The philosophy of proprietary software implies that the best way to drive innovation is to make it lucrative by linking revenue to progress. Setting up a business from a product ensures that developers are committed to improving the product for their paying end users and creating continuous value (Thompson, 2020).

This long-lasting mindset was only called into question again by the emergence of the open-source business model. This is discussed in chapter 2.2.

2.1.2 Borderline

This subitem clarifies which properties proprietary software must necessarily have. But first the most relevant terms have to be clarified.

Proprietary software is not the equivalent of commercial software, although the two terms are sometimes used synonymously in articles about free software. Even software distributed free of charge - e.g., freeware - can be "proprietary" if the licensee cannot acquire the freedom of use as with open-source software. Even if this could only be verified by means of Internet research, it should be noted that proprietary software should be distinguished from commercial software. It is possible for software to be commercial (i.e., aimed at making a profit for the producers) without being proprietary. The opposite case, as mentioned with freeware software, is also possible (Raymond, 2015).

There are sufficient popular examples of closed-source software. To list them would go beyond the scope of this article. Hence a different approach: If a software is not open source, it is proprietary. Only then it is possible to distinguish between freeware and commercial software.

2.2 Open-source Software

Open-source software (OSS) has recently been of great interest both to the software industry and to economic theory. What used to be the strategy behind the operating system Linux has turned into a growing and much-studied phenomenon. Open-source has led to a rethinking in software development and has also established itself as an antithesis to closed-source software distribution (Dalle/Normale/Cachan et al., 2002). Many economic actors now decide for an open-source strategy, i.e., adopt open-source software products and even sometimes publish the sources of the programs they have written instead of keeping them for themselves as used to be the case in the usual proprietary model. The long-lasting hype about open-source has also led the established media to claim that open-source will be the future of software development (Learnings from Linux, n.d.; Noyes, 2013).

Reason enough to take a closer look at the topic in this chapter, with regard to the historical background and the clear differentiation from proprietary software.

To get the strategy behind the development type, open source requires a clear definition. By its nature, OSS is "free" which means that the source code (human readable code) is made publicly accessible at no cost, so any conceivably individual can download the source code, compile it into binary code (machine readable code), and run it on their computer (Sacks, 2015). Software can be made open source by individuals for altruistic motives as well as by organizations or companies.

The legitimate question that arises in this approach is: Why should so many individuals and whole communities dedicate to provide their work as open-source projects from which they seem to get no reward, while these developments provide great utility and thus create considerable monetary value?

To illustrate: The most popular OSS projects, the operating system Linux has a rapid growing market share of 3.6% in 2020 and the browser Firefox from the Mozilla foundation counted over 850 million downloads in 2018 (Henry, 2020; Hayon, 2019).

These two successful stories prove that open-source developments are not just peripheral phenomena but have been widely accepted for quite some time. In order to answer the question mentioned above, we need to know the principles and background of the motives for OSS.

2.2.1 Historical Background and Principles

Open source has a long and controversial history.

The first influences of the open-source Strategy were for instance the do-it-yourself movement, the hacker movement of the 1960/1970s and the Free Software movement of the 1980s (Singh, 2018; Aioma, 2019). At that time, software vendors began to increasingly copyright their technologies, withhold source code and demand licensed use of software for a fee. More precisely, several companies began to deliver software not in the previously common form of program source code, but in the form of a purely machine-readable format, the so-called binary format. Proprietary software took broad market share in this world of technology (Singh, 2018).

In contrast, a group around Richard Stallman at the Massachusetts Institute of Technology (MIT) Artificial Intelligence Lab founded an antithesis. According to his thesis, software should not have owners (Stallman, 1994).

As Stallman (1994) argues the society needs information that is truly available to citizens themselves - for example, programs that people can read, correct, adapt and improve on their own, not just operate. Nowadays, software developers only offer black boxes that cannot be viewed (Stallman, 2002). In his final part he claims that the society needs to encourage the spirit of voluntary cooperation in its citizens. And therefore "[…] free software is a matter of freedom, not price" (Stallman 1994, p. 26). In summary, Stallman was primarily concerned that users should be able to use the software at their own discretion and adapt it if necessary.

But the question about the motives still exists. Sufficient research has been done on the motivation behind the development of open-source. Based on the most popular OSS projects, two particularly relevant scientific literatures were conducted by Hertel, Niedner & Herrmann (2003) on Linux and Mendonca & Sutton (2008) who dealt with Mozilla. Both found certain points that every OSS project has in common.

In the studies mentioned, two general groups of motives are identified: intrinsic and extrinsic motives. While intrinsic motivation describes the situation in which somebody is doing something because it is inherently interesting, enjoyable or challenging, in the case of extrinsic motivation, someone expects a separable outcome (Ryan & Deci, 2000).

Reviewing the rapidly emerging literature on OSS, the three crucial themes which regularly appear when analyzing the intrinsic motivation of OSS programmers and in particular of initiators are (1) the need for a specific software solution, i.e. the phenomenon of user programmers, (2) the fun of play and (3) the desire to give a present to the programmer community, i.e. a gift benefit (Bitzer, Schrettl & Schröder, 2007). In their model of the "OSS provision game" the three authors explained the reasons why developers are more likely to provide their software open source.

Spaeth, von Krogh & He (2015) deliver more accessible motives in their paper "Perceived Firm Attributes and Intrinsic Motivation in Sponsored Open-source Software Projects". They report that young OS developers are driven by interest, fun, altruism, recognition within the group and willingness to learn. In addition, established OS projects offer an ideal springboard for further careers.

In recent years, this "innovation of the thousands" has also established itself in the private sector, especially at technology companies such as Google and IBM. They have recognized that open-source communities program hundreds of thousands of lines of code for a new product much faster and more efficiently than a closed, in-house team (Schlaefli, 2014).

This aspect will be discussed later on in chapter 5 and 6.

2.2.2 Borderline

In order to also find a delimitation, as in the more detailed definition of proprietary software, it is also necessary to discuss common synonyms in linguistic usage.

An analysis from Hars (2002) found that Open-source software is generally confused with trial software, free software, share-ware, or royalty-free binaries. Moreover, freeware is software that is completely free for anyone to use or pass along to others to use. With trial software, consumers can try the software before they decide to buy - they only have to pay when the trial period has expired (Dhir & Dhir, 2017). With shareware, it is the intention that copying and passing it on to others, as long as it is the trial version of the software and not the registered version (Hippel & Krogh, 2003).

Beyond all these aspects of various synonyms, open-source software stands out. The relevant distinguishing characteristics can be seen especially in terms of license rights, redistribution, accessibility and individual modifications.

3 Comparison

The division into these two very different development approaches have led to a split among software developers. The antitheses led to a far-reaching philosophical debate. On the one side there are open-source supporters insisting that the open-source philosophy fosters transparency and collaboration amongst developers, promoting faster technological advancement and innovation. For open-source fans, the principle is that the entire community, including non-programmers, benefits from developments and progress, which will encourage better software for everyone is justifiable (Thompson, 2020).

But for the business-minded side, open-source software simply does not make sense. It is understandable that the development of software involves a significant amount of effort, which ties up resources such as time, previous knowledge and organization. This effort should be monetarily compensated and the developer himself should be credited for the work done.

Taken together, these contrary views are indeed very logical. Thus, a deeper insight and a far-reaching comparison between closed and open-source development is appropriate.

To achieve this, the following listing (Table 1: Comparison PS and OSS) describes the three most common aspects of software development and how proprietary and open-source software influences them. The analysis is done from the viewpoint of the vendor and partly from that of the consumer.

Table 1: Comparison PS and OSS

Aspect	**Proprietary software**	**Open-source software**
Product quality	For an increase in cost, stability, more commercial support, and software development, people tend to move to proprietary solutions and that aspects are more covered by closed-source projects (Dhir/Dhir, 2017).	Dhir & Dhir (2017) found out for operating systems that in terms of the long run open-source software will have higher general quality than commercial code software in terms of security, free support, compatibility and availability. This also applies to mobile OS industry and the web browser industry.
Usability	Some sources claim that proprietary and commercial software developers deliver an improved overall user experience. Commercial vendors rely on the customer choosing their solutions over the number of free, open-source alternatives on the market. User interfaces are therefore usually sleeker, and general usability is often on a higher standard in a closed source product (Thompson, 2020). Particularly when it comes to user-friendliness and functions, software companies, which are constantly exposed to free-	Open-Source Software has a reputation for being not very user-friendly and therefore hardly suitable for normal users. According to Mühlig (2005), this is mainly due to the fact that the most common open-source technologies focus more on the server and backend-side than on the desktop. Furthermore, in the long to mid-term, OSS cannot exploit user-friendliness as a market advantage.

	market competition, must constantly improve.	
User dependency	As proprietary software is privately-owned, this indicates that the user has no control over the lifespan and the technical support. In addition, there is a potential risk that when a primary developer drops out of support, the already integrated software will become unusable (Kazmeyer, n.d.). With the vendor lock-in strategy (or the Pottersville pattern), this targeted approach has already reached parts of the overall market. In simple terms, the aim is to bind customers to the software in the long term and to discourage them from changing provider by incurring substantial switching costs (Mizinska, 2019).	The distributor of Linux, Red Hat, states on their corporate website that open source offers not only more flexibility, but also lasting longevity (Red Hat, n.d.). In summary, open-source guarantees a minimization of customer dependency due to longer development and accessibility. In order to avoid the vendor lock-ins mentioned for proprietary software an empirical survey from January 2020 revealed that 62% of respondents use open-source software for exactly this reason (Pescador, 2020).

Overall, these results indicate that the selection of proprietary or open-source software is characterized by different trade-offs. The decision is up to the individual, who weights certain aspects according to personal preferences. In section 6 additional properties are listed, as well as further factors of the selection process within companies when choosing software.

4 Proprietary and Open-source Markets in a Business Context

Popular media indicate that OSS is the future and will prevail against their counterparts proprietary software in the long run (Volpi, 2019; Noyes, 2013).

Sacks (2015) argues that these forecasts are irrelevant for the tech industry. According to him, developers of proprietary and open-source software would not compete in the same market, even if there were overlaps in certain target markets. However, the underlying nature of the open-source development process leads to a self-selection process in which proprietary source developers differentiate their product in order to focus on the market they create, which is not the target of OSS. Thus, the vendors of proprietary software are in tougher competition with each other than with the open-source software vendors. It should therefore be assumed that both development approaches will continue to coexist in the long term as long as the underlying markets targeted by the two methods continue to exist (Sacks, 2015).

To understand these markets, it is necessary to understand the product itself at first. Software is perceived by many individuals as a product that can be bought and owned just like any other physical good. However, software is more than just owning a copy of a product that can be used legally (Christl, 2008). This statement also helps with another aspect: In discussions about open-source software development, the following question is often raised: How is it still possible to earn money with that, although the source code is made freely available to everyone. Vendors of proprietary software like to create the illusion that the offered software is commercial, conventional and tradeable. Thus, the character of a physical good is suggested, even if software is bound to associated hardware. Many other examples exist of how software can be "personalized" if it is originally only a copy of a unique set of computer-readable instructions, access to which is granted by means of a unique (license) key. These basic concepts (coming from the proprietary niche) are softened by open-source developers. The collaborative (partly free of charge) production of source code, the continuous improvement and provision of this software is in strong contrast to that of proprietary software. Dealing with OS, the main sources of income, besides sales, are long-term support, consulting and deployment. (Christl, 2008).

To be aware of these opposites, the following table (Table 2: Economic aspects) highlights and describes selected aspects for economic analysis.

Aspect 1: Total cost of production. From a software vendor's perspective, the total cost of production in the short term are a useful starting point for dividing total costs into two categories: fixed costs that cannot be changed in the short term and variable costs that can be changed in the short term (Khan Academy, n.d.). A common method is to keep part of the hardware and software development in-house and outsource or contract out the rest. What cause the companies to do this and how this is done is explained in the following table (Table 2: Economic aspects) (Pighin, 2013).

Aspect 2: Total cost of ownership. The term Total Cost of Ownership (TCO) was first used in 1987 to help buyers determine both the direct and indirect costs of a system. Within software development, the term is understood to include the costs of developing, improving, maintaining and supporting an application (PSL Corp, n.d.). Even if one is strictly speaking only the "holder" of the software and not the “owner”, this concept is often used to compare the costs of different software options and helps in the selection process.

Even if proprietary and open are only production models, certain approaches of a underlaying business model are assumed.

Table 2: Economic aspects

Aspect	Proprietary software	Open-source software
Total cost of production (TCP)	Campbell-Kelly and Garcia-Swartz (2010) classified two types of investments in the production process of software development, which are common in proprietary companies: (1) Investments in R&D and (2) acquisitions. The latter is used to promote in-house software development, where the related know-how of often smaller companies is integrated into development process.	In general research, the focus in economic analysis is understandably mainly on open source. The assumption in the early emergence of open-source projects, that talents, due to the large community involved, are freely available, is no longer valid. Nowadays, OS developers make it partly similar to PS vendors (as described). Leading OSS companies, e.g., Red Hat, are constantly acquiring smaller companies in order to use both the software itself and the software-producing team of the acquired company and to integrate them into their own team (Campbell-Kelly et al., 2010). The counter-financing of this venture does not necessarily have to come from the sale of licenses. Cost recovery for creating and maintaining open-source software can be achieved by selling services such as training, technical support or consulting (Germain, 2013).

Total cost of ownership (TCO)	When it comes to TCO, Lin (2008) sees the main point of purchase cost of proprietary software as the decisive factor from a company's point of view. Even though companies may have different levels of qualification with regard to proprietary software, all companies trust the manufacturer for new versions and upgrades. An example: With an increasing number of servers, proprietary solutions become more and more expensive. First, many proprietary systems (including Microsoft) sell pro-client licenses; this means that even if the hardware can support more clients, one must pay more to actually use the purchased hardware. Second, one has to pay more for proprietary systems if there is a need for more computers (FOSS technologies, n.d.).	A now outdated study suggests that the answer to the question of lower costs distributed over the life cycle of proprietary or open-source solutions is not easy to give (Thomas, 2004). With the adaptation of an open-source software solution, the costs seem lower at first glance. From case to case one “saves" license fees, which allow "try before you buy" (Roy, 2006). At second glance, however, it becomes clear that this approach does not go far enough. With the implementation of open-source into the corporate IT-ecosystem many sunk costs are incurred. Lin (2008) provides empirical evidence that the total cost of open-source software deployment varies from company to company and depends largely on the skills and expertise of the IT staff in an organization. Thus, if a company's IT staff actively contributes to an open-source project, the support costs for this software are intuitively very low.

		An example (continued): In contrast to the proprietary solution mentioned above, most GNU/Linux distributions allow the installation of an unlimited number of copies at no additional cost and there is no performance limit built into the software. There may be a fee for additional support from the manufacturers and more trained personnel may be needed (FOSS technologies, n.d.).

The comparison of these results with those of other studies confirms that the integration of proprietary software does not necessarily have to be more expensive for the customer than open-source software. Especially commercial open-source solutions have lower acquisition costs but require professional (and therefore inevitably expensive) support, either internally or externally (Ahmed, 2020).

As selection criteria the above-mentioned aspects, which should naturally be weighted differently from company to company and adapted to individual needs, should be used. A general statement on this matter is therefore difficult to make.

5 Proprietary and Open Software in the Big 3 Techs Companies

The acceptance of open source or the commitment to the proprietary software approach can best be analyzed at the leading tech companies. Apple, Microsoft and Google play a dominant role in their markets as well as their behavior has an effect on the attitudes of small and medium-sized companies. This section deals with the approach to developing software and its distribution.

5.1 Apple

When Apple announced in late 2015 that its programming language Swift will be available as open source, the developer community was rightly pleased. At the same time, the Apple's developer-website pompously stated that Apple was "the first major company to make Open Source development a key part of its software strategy, continues to use and release significant quantities of open source software" (Steven Vaughan-Nichols, 2015). This statement has disappeared from the current official website for undetermined reasons. Vaughan-Nichols (2015) explains his concerns about this claim in a blog post from ZDNET and even describes it as "ill-thought out". The elaborations from the previously mentioned blog post and an article in PM Network suggest that, although Apple has been using the open source approach for years, they were the first major company to take advantage of it for monetary gain and subsequently for profit (Rockwood, 2016).

The thesis of S. Vaughan-Nichols (2015) stating that Apple is relatively slow with the publication of the source code can also be proven by the operating system "Big Sur" released in 2020. On the website, where the operating systems macOS are offered for download, exactly the latest version number 11.0 is missing (General availability: November 12, 2020; observation date: December 17, 2020, https://opensource.apple.com).

To clarify the controversy described above, it is still necessary to determine why Apple presents itself as a true open-source vendor. From a business or advertising point of view, the company naturally wants to polish up its image and present itself as a company that clearly identifies with the open-source developer community (Asay, 2016). However, there is no significant scientific evidence from the product-specific side. On the internet portal Quora Apple's strategy is seen in the fact that it is pure a tactic to provide the source code of some projects open source in order to save maintenance by patches and other support services (Lambert, 2016).

Even though Apple generates most of its revenue from hardware sales, there are also several proprietary software solutions in the product portfolio. For example, the IT company uses proprietary diagnostic tools software to limit third-party repair of MacBooks. On the one hand, the goal is to ensure security on the part of Apple, on the other hand, critics claim that the company is trying to control the market for repairs and that customers are therefore driven to buy completely new products (Statt, 2018).

5.2 Microsoft

As already described in paragraph 2.2.1 about the historical background of open source, in the late 1970s / early 1980s computer manufacturers began to keep their source code closed and to charge a fee for access. The beginning of the era of proprietary software was initiated. In this environment, Microsoft became a successful pioneer of the proprietary and commercial development and set up the distribution model for software without hardware. As of today, nearly all of Microsoft's software is proprietary, including the Windows operating system family and Microsoft Office (“Proprietary Software Definition", n.d.).

To understand Microsoft's commitment to proprietary and commercial software development at that time, there should not be asked for the reasons for choosing the proprietary way. Rather, why not open source? At that time, the open-source strategy was seen as a threat to their business model by the Microsoft CEOs, represented here by Bill Gates and Steve Ballmer (Niu, 2019). As already described in the comparison, when distributing proprietary software, large companies tend to provoke a certain dependency on their customers - the so-called vendor lock-in effect. This requires an enormous management effort, strategic game-play, patenting and branding, and flashy product launch events (Ferguson, 2005). The story of Windows Vista is a very striking example of the monetary dimensions involved in a non-market-ready development. According to the American business magazine Bloomberg, around 10,000 employees were simultaneously working on this project, which cost about ten billion US dollars (Ricadela, 2008).

Microsoft's drastic attitude towards open source has changed gradually in recent years. Since the former CEO Steve Ballmer described Linux as "a cancer that attaches itself in an intellectual property sense to everything it touches" (Greene, 2001), the company has changed direction so that they recently admitted it was wrong about open source (Warren, 2020).

By becoming the largest single contributor to projects worldwide, beating Facebook, Docker, Google, Apache and many others, Microsoft today is a big player in the open-source software area. Microsoft has also gradually introduced the in-house open-source strategy in recent years, including open sourcing PowerShell (Calvo, 2016) ,Visual Studio code (VScodium, n.d.) and even the original JavaScript engine from Microsoft Edge (Gaurav & Adalberto, 2015).

By overtaking GitHub, a large code repository mainly for open-source developers Microsoft seems to have put aside its indifference with open source (Warren, 2018).

Even market-dominant companies such as Microsoft, which have been reluctant to open their proprietary innovations to users and competitors in the past, have recognized that they must coexist with the open-source innovation model in order to be successful (Rao, Klein & Chandra, 2011).

5.3 Google

Besides Microsoft, Google is also emerging as a big player in the field of open source software development (Asay, 2017).

Google is a major user of open-source software both in its internal systems and in its online services, which provide Google with substantial revenues from Internet advertising. As such, Google has therefore a strong incentive to contribute to OSS innovation. Google thus fits well with Von Hippel's (2006) definition of lead users as members of a user population that has two distinguishing features: (1) they are ahead of an important market trend and currently working on solutions long before many users need them; (2) they expect a relatively high benefit when they get a solution for their needs, and therefore they innovate. For this reason, Google also takes a leading role in the Open Handset Alliance organization, along with other leading tech companies and mobile carriers. They are committed to promoting an open Linux-based platform that provides the development of diverse but compatible mobile phone applications. Google calls this platform "Android". The objective is to accelerate the ability of customers to use the internet for software, content and services on the smartphone in the same way as they do from their PCs (Rao et al., 2011).

Just like the PC operating systems, it took a long way for mobile operating systems to get to get their current form. Important for the differentiation at the abundance of mobile operating systems is the availability of the option to use third-party applications. The mobile operating system Android initiated by Google guarantees exactly this level of commitment, together with middleware and key applications for use on mobile devices (Dhir et al., 2017). With a 2018 market share of over 85% of the global mobile operating system market, Android also strengthens Google's main business, the advertising business (Ruchi, 2018).

With Chrome, Google also has the most popular browser worldwide in its proprietary product portfolio, which is based on the open-source project Chromium. Google developers add their own proprietary code to Chromium, which creates services such as the browser's automatic update mechanism and features like the tabbed user experience to create the actual Chrome browser and to border with it from other Chromium-based competitors like Opera etc., what can be considered a competitive advantage (Keizer, 2020).

Based on the existing literature and the latest research, it is clear that Google has managed to cleverly combine the advantages and disadvantages of open source and proprietary development approaches.

6 The Distribution of Proprietary and Open-source Software

In information technology, the relevance of different concepts is often measured by how widely spread and established certain products are. To this sense different markets are specified in the following section, in order to work out, how the proprietary and open-source concepts are used within it. Existing research recognize the crucial role that the computer operating system market, the mobile operating system market and the web browser market play in this meaning.

Computer operating system market. The market for computer operating systems has always been dominated by Microsoft and its proprietary product Windows. According to Statcounter, almost three quarters of all PCs worldwide have a version of Windows installed (StatCounter Global Stats, 2020b). Despite the mostly proprietary access to Microsoft's operating system, there are also some open-source approaches, such as providing a complete Linux kernel for the Windows subsystem (Ostler, 2020). This also leads to the most prominent representative of the open-source software movement: Linux. With a market share of about 2%, it is not the main competitor of Windows, but in its development, it follows a completely different path (StatCounter Global Stats, 2020b). Linux is offered as open-source software and allows the free code to be viewed, edited and - for those with certain skill-sets - to contribute it (Dhir/Dhir, 2017).

Mobile operating system market. In contrast to the market segment mentioned above, open-source technology is more strongly represented here. In this context a mobile OS (operating system) runs on smartphones, tablets or as a digital assistant. It is also crucial to determine whether third-party applications are accepted and allowed by the respective OS. The second most widespread participants in this market are understandably contrary in their philosophy. While Apple's IOS is a proprietary product with single open-source components, Android has a completely open code (Dhir/Dhir, 2017). Other than on the computer OS market, the open-source representative has had a dominant position in this market since 2013 (StatCounter Global Stats, 2020c).

Web browser market. The important market for the supremacy as a browser has always been considered highly competitive. No less than six providers are in direct competition with each other. Even if the market share is "only" about 4% in 2020, Mozilla's Firefox browser follows an interesting course (StatCounter Global Stats, 2020a). Firefox was designed for simplicity, security, and extensibility and this was largely made possible by its open-source concept. This was also the big success. Firefox was the long-time competitor of Microsoft's proprietary Internet Explorer, until Google's Chrome, which (as mentioned in section 5.3) has open-source elements of its own, became widely accepted in 2012.

The above-mentioned markets relate primarily to private customer business. The spread of proprietary and open-source software in companies as enterprise software is highlighted by Vaughan-Nichols (2015) in his article. The result of his interpretation of a study of an open-source software logistics and service provider is quite interesting. According to this study, open-source software is already used by a majority of all companies. However, according to the survey, companies still have a lack of formal OSS management guidelines. Additionally there is something like "blind trust" in the further development of open-source business software (Vaughan-Nichols, 2015). More recent studies, especially those commissioned by the software company Red Hat, come to a not so clear, but similar result. It should be noted that the choice between the concepts goes beyond the cost factor. Organizations that use enterprise open source see a variety of benefits: The IT executives surveyed stated that the higher quality of software was the main reason they chose open source over proprietary, followed by better security and lower total cost of ownership (Red Hat, 2020).

All in all, the open-source concept has long since arrived at the broad masses and in some areas (as described business software) displaces its counterpart.

7 Conclusion and Discussion

Although in early years open source was considered only a niche, the concept of free source code and unrestricted customization became established in the mainstream. It has now become an important part of modern application development. Surveys show that across industries, more than half of mainstream companies are using or have plans to use open-source software in business. They run parallel to the proprietary solutions already implemented (Litzel, 2019). Leading IT companies are responding to this phenomenon by continuously integrating open-source elements into their software solutions. Despite all this, they partly stick to the proprietary mindset and make clever use of the advantages. Advantages result from an economic but also from a product-specific perspective. The division into two worlds of development model established in the 1980s and subsequent years has long since disappeared.

From today's perspective, the reality that there is an option to reveal source code to users has the potential to conquer their policies set by proprietary software and creates its very own specialized niche within the software ecosystem. Despite all this, software providers with different development strategies do not necessarily compete in the same market. The product offered does not understand itself as such and thus appeals to different target groups. Therefore, in the end it is irrelevant which form of development will prevail. In the long term, the advantages of each model complement each other, which suggests that they will coexist together for quite some time. For the consumer, this outcome is only advantageous: reasonable price and high degree of freedom. The decision criterion is a combination, a trade-off. The questions that will arise for software developers (without a clear inclination) in the future are which philosophy to be mainly preferred and to what degree: proprietary or open source?

8 Bibliography

Ahmed, Irfan (2020, April 29). *Commercial Proprietary Software vs Open Source - Which is Better?* https://www.astera.com/type/blog/why-proprietary-software-can-be-more-cost-effective-than-open-source/, accessed December 10, 2020

Aioma.com (2019, March 1). *Die Geschichte und Zukunft von Open-Source Software*. https://www.aioma.com/de/blog/open-source-software-geschichte-zukunft, accessed October 30, 2020

Asay, Matt (2016, November 9). *Apple is doubling down on open source*. https://www.techrepublic.com/article/apple-is-doubling-down-on-open-source/, accessed November 25, 2020

Asay, Matt (2017, October 30). *Why Microsoft and Google are now leading the open source revolution*. https://www.techrepublic.com/article/why-microsoft-and-google-are-now-leading-the-open-source-revolution/, accessed November 26, 2020

Bitzer, Jürgen/Schrettl, Wolfram/Schröder, Philipp J. H. (2007). Intrinsic motivation in open source software development. *Journal of Comparative Economics*, *35*(1), 160–169. DOI: 10.1016/j.jce.2006.10.001

Calvo, Angel (2016, August 17). *Windows PowerShell is now "PowerShell": An Open Source Project with Linux support – How did we do it?* https://devblogs.microsoft.com/powershell/windows-powershell-is-now-powershell-an-open-source-project-with-linux-support-how-did-we-do-it/, accessed November 26, 2020

Campbell-Kelly, Martin/Garcia-Swartz, Daniel D. (2010). The Move to the Middle: Convergence of the Open-Source and Proprietary Software Industries. *International Journal of the Economics of Business*, *17*(2), 223–252. DOI: 10.1080/13571516.2010.483091

Christl, Arnulf (2008). Free Software and Open Source Business Models. In Hall, G. Brent/Leahy, Michael G. (Eds.), *Open Source Approaches in Spatial Data Handling* (21–48). Berlin, Heidelberg: Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-74831-1_2

Dalle, Jean-michel/Normale, Ecole/Cachan, Suprieure/Jullien, Nicolas (2002). Open-Source vs. Proprietary Software.

Dhir, Swati/Dhir, Sanjay (2017). Adoption of open-source software versus proprietary software: An exploratory study. *Strategic Change*, *26*(4), 363–371. DOI: 10.1002/jsc.2137

Ferguson, Charles (2005). How Linux Could Overthrow Microsoft. *Technology Review*, *108*(6), 64–69.

Finkel, Lonnie (2017, December 7). *How To Protect Your Company's Software Assets*. https://finkellawgroup.com/protect-company-software-assets/, accessed November 5, 2020 fosstechnologies.weebly.com (n.d.). *Total Cost Of Ownership (TCO)*. http://fosstechnologies.weebly.com/total-cost-of-ownership-tco.html, accessed December 9, 2020

Gaurav, Seth/Adalberto, Foresti (2015, December 5). *Microsoft Edge's JavaScript engine to go open-source*. https://blogs.windows.com/msedgedev/2015/12/05/open-source-chakra-core/, accessed November 26, 2020

Germain, Jack M. (2013, November 5). *FOSS in the Enterprise: To Pay or Not to Pay?* https://linuxinsider.com/story/foss-in-the-enterprise-to-pay-or-not-to-pay-79341.html, accessed December 9, 2020

Greene, Thomas C. (2001, June 2). *Ballmer: "Linux is a cancer."* https://www.theregister.com/2001/06/02/ballmer_linux_is_a_cancer/, accessed November 26, 2020

gs.statcounter.com (2020a). *Browser Market Share Worldwide*. https://gs.statcounter.com/browser-market-share, accessed December 3, 2020

gs.statcounter.com (2020b). *Desktop Operating System Market Share Worldwide*. https://gs.statcounter.com/os-market-share/desktop/worldwide, accessed December 10, 2020

gs.statcounter.com (2020c). *Mobile & Tablet Operating System Market Share Worldwide*. https://gs.statcounter.com/os-market-share/mobile-tablet/worldwide/, accessed December 3, 2020

Hars, Alexander/Ou, Shaosong (2002). Working for Free? Motivations for Participating in Open-Source Projects. *International Journal of Electronic Commerce*, *6*(3), 25–39. DOI: 10.1080/10864415.2002.11044241

Hayon, Dominik (2019, August 29). *Mozilla legt offizielle Zahlen vor: So viele Menschen nutzen Firefox*. https://www.chip.de/news/Mozilla-legt-offizielle-Zahlen-offen-So-viele-Menschen-nutzen-Firefox_147510433.html, accessed November 27, 2020

Henry, Matthew (2020, July 24). *Linux market share grows 210% in 5 months*. https://www.esop.pt/en/destaque/linux-market-share-grows-210-5-months, accessed November 27, 2020

Hertel, Guido/Niedner, Sven/Herrmann, Stefanie (2003). Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. *Research Policy*, *32*(7), 1159–1177.

Kazmeyer, Milton (n.d.). *Disadvantages of Proprietary Software*. https://smallbusiness.chron.com/disadvantages-proprietary-software-65430.html, accessed November 13, 2020

Keizer, Gregg (2020). Google's Chromium browser explained: Chrome is the most popular browser in the world, but there would be no Chrome without Chromium, the open-source project that underpins it. Here's what Chromium is, where you can download it, how it shapes your online experience, and -- if you hate it -- how to get rid of it. *Computerworld (Online Only)*, 1–1.

khanacademy.org (n.d.). *The structure of costs in the short run (article)*. https://www.khanacademy.org/economics-finance-domain/microeconomics/firm-economic-profit/average-costs-margin-rev/a/the-structure-of-costs-in-the-short-run-cnx, accessed December 10, 2020

Lambert, Terry (2016, August 9). *Why does Apple open-source some of their projects?* https://www.quora.com/Why-does-Apple-open-source-some-of-their-projects, accessed November 25, 2020

Lin, Lihui (2008). Impact of user skills and network effects on the competition between open source and proprietary software. *Market Transformation in a Networked Global Economy*, *7*(1), 68–81. DOI: 10.1016/j.elerap.2007.01.003

Litzel, Nico (2019, September 3). *Open-Source-Software verbreitet sich zunehmend*. https://www.bigdata-insider.de/open-source-software-verbreitet-sich-zunehmend-a-860469/, accessed December 10, 2020

Mendonca, Lenny T./Sutton, Robert (2008). An interview with Mitchell Baker. *McKinsey Quarterly*, (2), 12–13.

Mizinska, Marta (2019, December 18). *5 Vendor Lock-In Strategies for Your Online Business*. http://straal.com/5-vendor-lock-in-strategies-for-your-online-business/, accessed November 13, 2020

Mühlig, Jan (2005). Open Source und Usability. *Open Source Jahrbuch 2005*, 87.

neuron.csie.ntust.edu.tw (n.d.). *Proprietary Software Definition*. http://neuron.csie.ntust.edu.tw/homework/94/ComputerIntro/Homework1/B9415002/propdef.htm, accessed November 26, 2020

Niu, Evav (2019, January 19). *19 Years After Bill Gates Stepped Down as Microsoft CEO, These 2 Successors Have Left Their Marks*. https://www.fool.com/investing/2019/01/19/19-years-after-bill-gates-stepped-down-as-microsof.aspx, accessed November 26, 2020

Noyes, Katherine (2013, April 17). *Open source is taking over the software world, survey says*. https://www.pcworld.com/article/2035651/open-source-is-taking-over-the-software-world-survey-says.html, accessed October 29, 2020

os2hq.com (n.d.). *Learnings form Linux*. http://www.os2hq.com/archives/linmemo1.htm, accessed October 29, 2020

Ostler, Ulrike (2020, July 6). *So viel Open Source steckt in Windows*. https://www.datacenter-insider.de/so-viel-open-source-steckt-in-windows-a-945359/, accessed December 3, 2020

Pescador, Rachel (2020, January 27). *Can Open Source Software Save You From Vendor Lock-in?* https://www.percona.com/blog/2020/01/27/can-open-source-software-save-you-from-vendor-lock-in/, accessed November 13, 2020

Pighin, Anthony (2013, May 23). *Software Development: Fixed Cost or Opportunity Cost?* https://www.embedded-computing.com/embedded-computing-design/software-development-fixed-cost-or-opportunity-cost, accessed December 10, 2020 pslcorp.com (n.d.). *Optimizing the Total Cost of Ownership on an outsourcing software development project - Blog - PSL Corp*. https://www.pslcorp.com/software-development/optimizing-total-cost-of-ownership-on-an-outsourcing-software-development-project/, accessed December 10, 2020

Rao, PM/Klein, Joseph A./Chandra, Ramdas (2011). Innovation without property rights and property rights without innovation: recent developments in the ICT sector. *Advances in Competitiveness Research*, *19*(1/2), 83–99.

Raymond, Eric S. (2015, April 8). *The Jargon File*. http://www.catb.org/~esr/jargon/html/P/proprietary.html, accessed November 12, 2020

redhat.com (2020, February 17). *The State of Enterprise Open Source 2020: Enterprise open source use rises, proprietary software declines*. https://www.redhat.com/en/blog/state-enterprise-open-source-2020-enterprise-open-source-use-rises-proprietary-software-declines, accessed December 10, 2020

redhat.com (n.d.). *What is open source?* https://www.redhat.com/en/topics/open-source/what-is-open-source, accessed November 13, 2020

Ricadela, Aaron (2008). Microsoft: What Cost the Vista Fiasco? *Bloomberg.Com* of June 11, 2008. https://www.bloomberg.com/news/articles/2008-06-11/microsoft-what-cost-the-vista-fiasco-businessweek-business-news-stock-market-and-financial-advice, accessed November 25, 2020

Rockwood, Kate (2016). Open for Business. *PM Network*, *30*(4), 12–13.

Roy, Rick (2006). Open Source's Mainstream Future. *Insurance & Technology*, *31*(2), 37.

Ruchi, Gupta (2018). *Google Android Will Remain on Top for a Long Time*. https://marketrealist.com/2019/08/googles-android-will-remain-top-long-time/, accessed November 27, 2020

Ryan, Richard M./Deci, Edward L. (2000). Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary Educational Psychology*, *25*(1), 54–67.

Sacks, Michael (2015). Competition Between Open Source and Proprietary Software: Strategies for Survival. *Journal of Management Information Systems*, *32*(3), 268–295. DOI: 10.1080/07421222.2015.1099391

Schlaefli, Samuel (2014, December 9). *Die Motivation der Open-Source-Community*. https://ethz.ch/de/news-und-veranstaltungen/eth-news/news/2014/12/die-motivation-der-open-source-community.html, accessed October 30, 2020

Singh, Vivek (2018, January 10). *A Brief History Of Open Source*. https://medium.com/gitcoin/a-brief-history-of-open-source-3928cb451767, accessed October 30, 2020

Spaeth, Sebastian/von Krogh, Georg/He, Fang (2015). Research note—perceived firm attributes and intrinsic motivation in sponsored open source software projects. *Information Systems Research*, *26*(1), 224–237.

Stallman, Richard (2002). *Free software, free society: Selected essays of Richard M. Stallman*. Lulu. com.

Stallman, Richard M. (1994). Why Software Should Not Have Owners., *Technos: quarterly for education&technology.*(vol. 3), 24–26.

Statt, Nick (2018, October 4). *Apple is using proprietary software to lock MacBook Pros and iMac Pros from third-party repairs*. https://www.theverge.com/2018/10/4/17938820/apple-macbook-pro-imac-pro-third-party-repair-lock-out-software, accessed November 25, 2020

Thomas, Daniel (2004). Study finds Linux has higher total cost of ownership thanWindows. *Computer Weekly*, 18–18.

Thompson, Andrew (2020, June 18). *What is proprietary software and how does it work?* http://entrepreneurhandbook.co.uk/proprietary-software/, accessed November 6, 2020

Vaughan-Nichols, Steven (2015, December 8). *Was Apple the first major open-source company? Not even close*. https://www.zdnet.com/article/apple-was-the-first-major-open-source-company/, accessed November 25, 2020

Vaughan-Nichols, Steven J. (2015, April 16). *It's an open-source world: 78 percent of companies run open-source software*. https://www.zdnet.com/article/its-an-open-source-world-78-percent-of-companies-run-open-source-software/, accessed December 3, 2020

Volpi, Mike (2019, January 12). *How open-source software took over the world | TechCrunch*. https://techcrunch.com/2019/01/12/how-open-source-software-took-over-the-world/?guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAACfMK8yJC01zY0ZbxBzIuZvr2IK-UjnODNj8co6NIXw2ksuTqiAvo-Xq3j6gfXpyRc2tmMrdWEEzEvooa96_LpBJvlhKE7fbiOz0o3F6e1PCKUWpr83r7J9ZLBJTQSXwN_MRjaysaIEi97tXGNo5_DMQJml625sbN_0VS6nbih_6&guccounter=2, accessed December 14, 2020

Von Hippel, Eric (2006). *Democratizing innovation*. the MIT Press.

vscodium.com (n.d.). *VSCodium - Open Source Binaries of VSCode*. https://vscodium.com/, accessed November 27, 2020

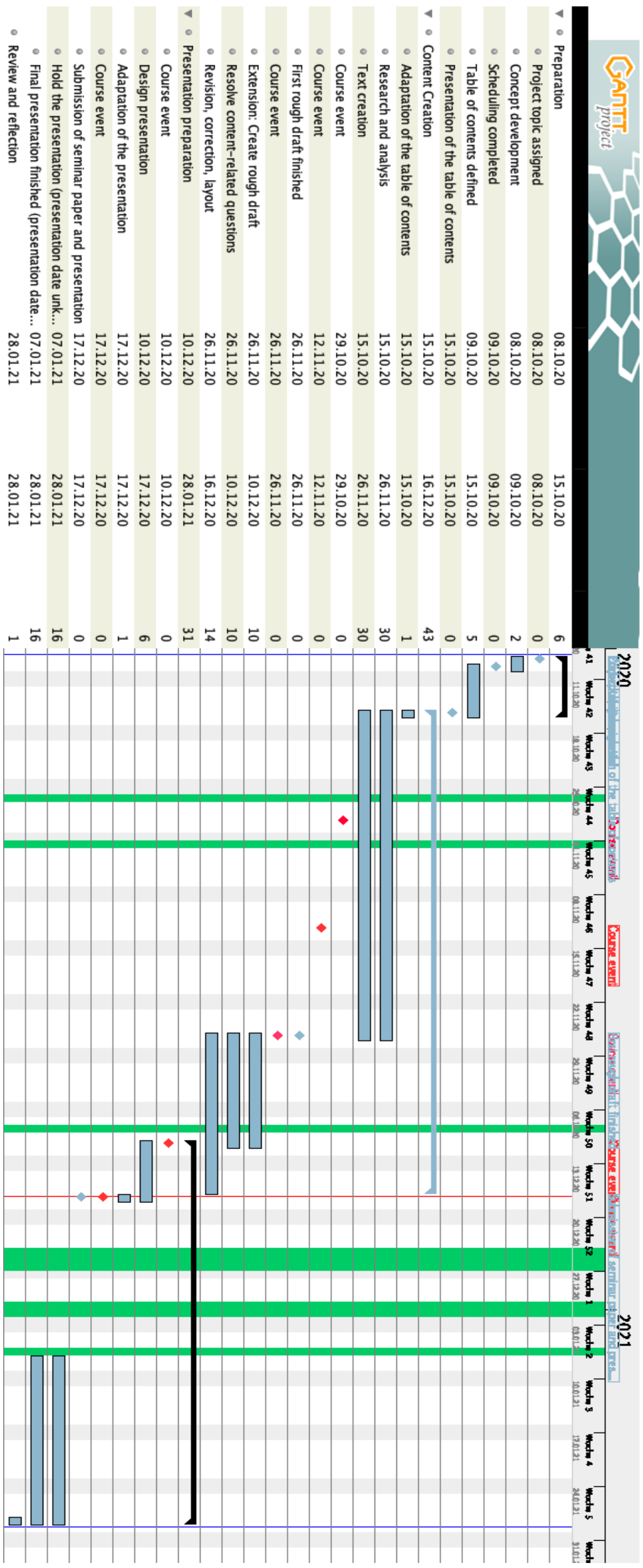
Warren, Tom (2018, October 26). *Microsoft completes GitHub acquisition*. https://www.theverge.com/2018/10/26/17954714/microsoft-github-deal-acquisition-complete, accessed November 26, 2020

Warren, Tom (2020, May 18). *Microsoft: we were wrong about open source*. https://www.theverge.com/2020/5/18/21262103/microsoft-open-source-linux-history-wrong-statement, accessed November 27, 2020

WebCampus (2017, September 18). *Open Source vs. Proprietäre Systeme - Was ist das Richtige für mich? - WebCampus - E-Learning Komplettlösung*. https://www.webcampus.de/blog/142/open-source-vs-proprietaere-systeme-was-ist-das-richtige-fuer-mich, accessed November 5, 2020

9 Appendix

(Gantt chart from October 8, 2020 to January 28, 2021)



Preparation	08.10.20	15.10.20	6
Project topic assigned	08.10.20	08.10.20	0
Concept development	08.10.20	09.10.20	2
Scheduling completed	09.10.20	09.10.20	0
Table of contents defined	09.10.20	15.10.20	5
Presentation of the table of contents	15.10.20	15.10.20	0
Content Creation	15.10.20	16.12.20	43
Adaptation of the table of contents	15.10.20	15.10.20	1
Research and analysis	15.10.20	26.11.20	30
Text creation	15.10.20	26.11.20	30
Course event	29.10.20	29.10.20	0
Course event	12.11.20	12.11.20	0
First rough draft finished	26.11.20	26.11.20	0
Course event	26.11.20	26.11.20	0
Extension: Create rough draft	26.11.20	10.12.20	10
Resolve content-related questions	26.11.20	10.12.20	10
Revision, correction, layout	26.11.20	16.12.20	14
Presentation preparation	10.12.20	28.01.21	31
Course event	10.12.20	10.12.20	0
Design presentation	10.12.20	17.12.20	6
Adaptation of the presentation	17.12.20	17.12.20	1
Course event	17.12.20	17.12.20	0
Submission of seminar paper and presentation	17.12.20	17.12.20	0
Hold the presentation (presentation date unk...	07.01.21	28.01.21	16
Final presentation finished (presentation date...	07.01.21	28.01.21	16
Review and reflection	28.01.21	28.01.21	1