Friedrich-Alexander-Universität Erlangen-Nürnberg
Technische Fakultät, Department Informatik

NATHALIE SCHNELZER
BACHELOR THESIS

A MODEL OF OPEN SOURCE LICENSES

Submitted on 8 June 2020

Supervisor: Prof. Dr. Dirk Riehle, M.B.A., Michael Dorner, M. Sc
Professur für Open-Source-Software
Department Informatik, Technische Fakultät
Friedrich-Alexander University Erlangen-Nürnberg

Versicherung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

_____

Erlangen, 8 June 2020

License

_____

Erlangen, 8 June 2020

Abstract

An open source license specifies all rights granted and obligations imposed to the user of the open source software. However, researcher and practitioners are faced with a large diversity of open source licenses. This can lead to a lot time and effort spent on examining the different licenses. Mistakes can lead to legal consequences.
Therefore, we used a sample study approach and developed a model of open source licenses from the acquired data to overcome this diversity. Licenses can be categorized in strong, weak, and non-copyleft licenses. They were also compared regarding their permissions, conditions, and limitations. The scope of examined licenses was limited to the ten most used licenses according to the Black Duck KnowledgeBase. Over the past years, a trend could be recognized from strong to non-copyleft licenses.
The resulting model reflects the various aspects of open source licenses mentioned above. Therefore, it enables researchers to identify conflicting licenses and practitioners to handle multi-licensed open source projects. It also helps practitioners to choose a specific open source license that is fitting for the requirements of the respective project.

Contents

1 Introduction

1.1 Original Thesis Goals

Our goal is to develop a model of the ten most used open source licenses. We will identify open source license properties, constraints, and usage. For this, a sample study approach was used. The resulting model shall reflect those open source license variation points and, therefore, enables researchers and practitioners to understand used licenses and helps to choose a license for their own work.

1.2 Changes to Thesis Goals

No changes were made.

2 Research

2.1 Introduction

"Open source is everywhere" (Szulik, 2018). In a survey by Tidelift (2018) over 700 developers were asked whether their products have open source software dependencies. Over ninety percent of their applications depend on open source software components. There are no significant differences concerning where the developers are from or how old they are. These results show that everyone uses open source nowadays.

However, open source software cannot be used without a license. Publicly available source code is not automatically free to use, copy and modify. For each file, a license must apply which defines the rights of the user. It is therefore very important to know about open source licenses. (*No License*, 2020)

A lot of the publicly available software does not have an open source license. On GitHub, which is the globally most used platform for software development with 100M repositories (*Build Software Better, Together*, 2020), in 2015, only about 20% of the repositories were licensed (Balter, 2015). Sen et al. (2008) noted that there is a lack in literature about choosing a license from the perspective of developers. But not only developers have trouble in dealing with open source licenses. According to a survey from Bitkom e.V., 54% of 570 German companies who use open source software do not have an open source compliance process (a standardized approach to handle open source software). This shows that while the usage of open source software is now very common, evaluating the compliance is not (Gentemann & Termer, 2020).

Licensing can be very complex and is juridical important as it has legal implications. There are more than 200 unique licenses (A. Goldstein, 2019). They have different requirements varying from "do what the fuck you want" (*Do What the F*ck You Want to Public License*, 2020) to some that explicitly forbid embedding the software in proprietary programs (Free Software Foundation, 2020g). Especially big software projects depend on many different components, which may have several different licensing properties. All imposed obligations must be tracked and fulfilled (Jaeger & Metzger, 2016). Therefore, in this thesis, a model of open source licenses was created to gain an overview.

Licenses can be divided up into three main categories (Jaeger & Metzger, 2016; A. Schaaf, 2012; Sen et al., 2008). These categories describe under which license it is allowed to distribute a combined work or a derivate of the licensed software. This results in certain requirements (e.g. the whole distribution including the own work must be licensed under the given open source license). In addition to these, the licenses give several permissions, conditions, and limitations. Licenses can be compared with these properties. If many differently licensed components are used in one project, their respective licenses must be compatible (Jaeger & Metzger, 2016).

This thesis aims to create a model to give researchers and practitioners an overview of open source licenses. To achieve that, a sample study approach was applied. A systematic literature review was performed in order to gather all relevant information. Our model was not created based upon all licenses, but on the ten currently most used ones. They have been the same for the past ten years, although the ranking has changed. The data is based on the Black Duck KnowledgeBase that includes 2.5 million open source projects (J. Goldstein, 2017). Over 90% of them are covered by the top ten licenses. The top ten licenses are similar in other databases (Balter, 2015; A. Goldstein, 2020).

2.2 Related Work

As open source software is now very common (Szulik, 2018) there is on the one hand a lot of literature coverage of this subject. On the other hand, licenses are rarely discussed. In our systematic literature we found out that usually the focus lies on the economic advantages and opportunities for companies. However, there is some literature dealing with this topic.

Schaaf (2012) compared different open source licenses. An insight in the basics of open source licensing is included. The compared licenses are the GNU General Public License (GPL), the GNU Lesser General Public License (LGPL), the BSD License, and the Artistic License. Those are commonly used licenses, but only four of the ten most used ones. Nowadays, the MIT license is the most used one, but back in 2012, it was the GPL (see Figure 3). Schaaf also only focuses on the usage of open source software in companies. Moreover, it is written in German.

Jaeger & Metzger (2016) provide a very comprehensive legal framework of open source licenses. It includes almost all top ten licenses except for the Eclipse Public License and the ISC License. The focus is on legal aspects. It is also written in German and consists of more than 360 pages. Because license matters change over time, revised editions are published every four to five years.

There is also a very comprehensive work about open source licenses by Välimäki (2005). It is focused on how the use of intellectual property has changed due to open source. The history and economic of open source licensing are explained. Specific licenses are discussed, but not all of the top ten.

2.3 Research Question

The following research question arises from the problems presented in the introduction

How to get an overview about open source licensing?

To answer this question, we must answer the following questions:

What defines an open source license?

What different types of licenses exist?

Which licenses are the most used ones?

What are the differences between these licenses?

What aspects are to consider when using components with different licenses in one project?

Which decisions must be made when choosing a license?

2.4 Research Approach

To identify the ideal research strategy we followed the ABC of Software Engineering Research by Stol and Fitzgerald (2018). This framework categorizes eight research strategies regarding the obtrusiveness of the research and the generalizability of the results. For our analysis we took a universal approach so that the results can be appliable to all kinds of software. The license properties were assessed and compared without any variable modifications. Hence our research was not obtrusive. Therefore, we chose the strategy Sample Studies, which belongs to the “A” of the ABC. This indicates a high potential for generalizability. First, we looked at Open Source Licenses in general and then we focused on a specific and representable sample. We chose the sample licenses based on their usage frequency.

Concerning the research method, we decided on a systematic literature review. This method leads to an relatively unbiased and thorough result (Kitchenham & Charters, 2007). For structuring this process, we adhered to the specification of Kitchenham et al. (2009). It consists of three phases: Planning, conduction, and documentation of the review.

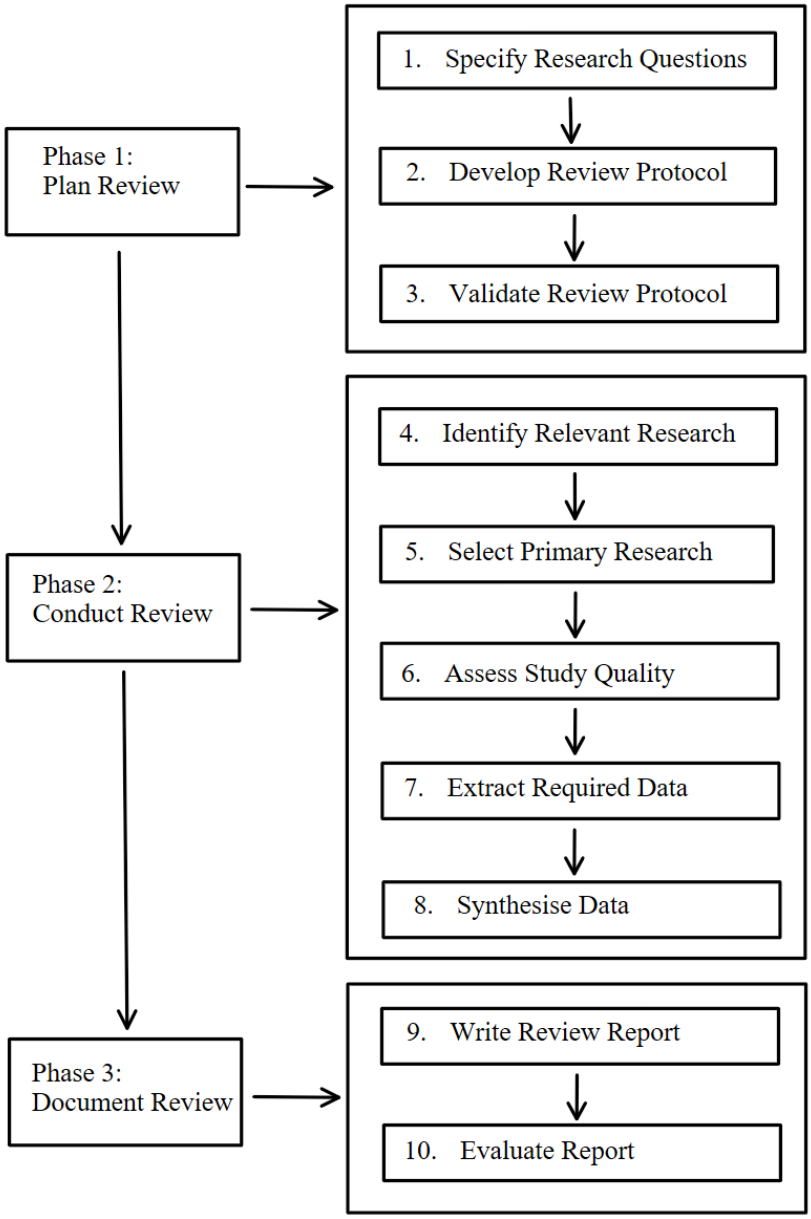

Figure 1. *Flowchart of a systematic literature review process.* Adopted from Brereton et al. (2007, p. 572).

As shown in Figure 1, the phases are subdivided into several steps. Phase one consisted of choosing the topic for the thesis and then specifying the research question. The overall question was then divided into multiple questions that are listed in chapter 2.3. These questions structure the research results. A review protocol was established to determine review conduction.
This includes the search process. The identification of relevant research was divided into three categories: searching electronic databases for general information, searching for license usage statistics, and searching the specific license texts. The study quality criteria were defined. A source was included, if it was relevant to the research, the language was English or German and the information was not outdated. We always aimed to find the most recent information. We did

not include research about the profitability of open source software or general advantages and disadvantages. We did not include community websites like stackoverflow.com because even though license matters are discussed frequently on there, the information published there is not reliable.
The required data was extracted through identifying the key statements an assigning the found literature to the specific research question. All found sources were added to Citavi, a reference management tool.

In Phase two the defined strategies were applied. We began by identifying relevant research. Concerning statistics about the usage frequency, the Black Duck KnowledgeBase was chosen as the primary source since it contains over 2.5 million open source projects and thus the most extensive collection of open source license data worldwide (J. Goldstein, 2017). For license texts, we searched the websites of the communities that develop and publish open source licenses. Most information about open source licenses can be found on the websites of the Free Software Foundation (FSF) and the Open Source Initiative (OSI). The FSF and OSI websites contain the license texts of all relevant licenses and give general advice on how to deal with open source software. Additionally, we searched on Google scholar, the online catalogue of the Friedrich-Alexander University, ScienceDirect, Springer Link and others. At first, we searched for "open source licenses" in general. We then adapted the search strings to the specific research questions, for example "license compliance" and "license compatibility". To identify relevant research, we also performed backward searching as described by Webster and Watson (2002). This means reviewing the references of identified sources for other relevant sources. As open source licensing only exists since about 30 years with the first GNU General Public License published in 1989 (Free Software Foundation, 2020q) and is continuously changing, we also included articles of different websites about the recent developments. Our primary web article sources were the WhiteSource and the Synopsis (the company who bought Black Duck) blog. Both are platforms for open source license and security compliance management. After this step the inclusion and exclusion criteria were applied.
The data was extracted, and the sources were added to Citavi. Sometimes one source contained information to answer several research questions. There we assigned every source to one or more outline points. To extract the data from the license texts, we specified several characteristics of licenses. We used Microsoft Excel to add the relevant license paragraphs to the characteristics and to compare the different licenses.

Phase three consisted of writing and evaluating the review.

2.5 Research Results

2.5.1 Definition of Free and Open Source Licenses

There are more than 200 different Open Source Licenses (A. Goldstein, 2019). Ninety-six of them are approved by the Open Source Initiative (OSI) (Open Source Initiative, 2020b) and 98 by the Free Software Foundation (FSF) (Free Software Foundation, 2020c). There is a slight difference between the terms free and open source software, as Richard Stallman explains on the FSF website in his article called "Why 'Open Source' misses the point of Free Software". According to Stallman, "Free Software" is about freedom. It supports social solidarity and is based on ethnic principles. In contrast, the OSI focuses on practical benefits and the technical advantages of making source code available. While the practical consequences are identical, the two concepts are based on different values and must therefore, be differentiated (Stallman, 2020). Therefore the terms FOSS meaning "Free and Open Source Software" or FLOSS for

“Free/Libre Open Source Software” are used to encompass both aspects. (Stallman, 2016) We will use the term OSS in this research because it is more commonly used.

A license is approved by the FSF as free software when it follows the four essential freedoms:

- The freedom to run the program as you wish, for any purpose (freedom 0).
- The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help others (freedom 2).
- The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

(Free Software Foundation, 2020e)

In short, an open source license must provide the freedom to use, study, copy, share, modify an distribute the program.

The OSI only approves a license, if it follows the Open Source Definition, which was originally derived from the Debian Free Software Guidelines (DFSG):

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost, preferably downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.

(Open Source Initiative, 2020c)

In short, a license must give anyone permission to use, modify, and share the software for any purpose. Some licenses only include these characteristics and add no further obligations. But there are also licenses which restrict the usage for example in terms of distribution but still are open source licenses.

Every creative work is automatically under exclusive copyright. It is not allowed for anybody else except the creator to copy, to modify or distribute it. This must be permitted explicitly. (*No License*, 2020)

The FSF stated, that only compiling or running a non-licensed software may infringe the copyright in some countries (Free Software Foundation, 2020c).

2.5.2 License Compliance

As there are many licenses with many different obligations it is important to assess them very carefully to detect and monitor possible risks. This process is called license clearance and if it is done correctly it eventually leads to license compliance. It is essential when a company or an individual uses open source software, because violating a license could lead to a copyright infringement or produce other legal conflicts. (Jaeger & Metzger, 2016)

To avoid such infringements the following steps should be considered:

- Analysis of the applicable open source licenses and determination of the license obligations

- Review of the implementation of the general obligations for the delivery of license texts, copyright notices, etc.

-Assessment of the compatibility of the different licenses

-Evaluation of the compliance with the copyleft

(Jaeger & Metzger, 2016, p. 23)

2.5.3 License Categories

To get an overview about all unique licenses it helps to categorize them. The distinction between strong, weak and no copyleft is the main criteria. (A. Goldstein, 2019) As described by the FSF, "copyleft is a general method for making a program (or other work) free (in the sense of freedom, not "zero price"), and requiring all modified and extended versions of the program to be free as well." (Free Software Foundation, 2020d).

Licenses are therefore described as permissive, if there is no copyleft clause included because relicensing under another license is permitted. (A. Goldstein, 2019; Salter, 2020). This specification was criticized by Phipps (2013). He argues, that describing non-copyleft licenses as permissive may lead to misunderstandings because these licenses do contain requirements and do not always permit everything. He also criticizes that the term copyleft may be confusing because it is not a well-known term. He therefore suggests using the term reciprocity instead because it catches the intuition of the community better. The copyleft concept determines in which way and to what extend the community wants the user to interact with them (Phipps, 2013).

To avoid misunderstandings, we chose not to use "permissive" as a license category. We did not let go of the term "copyleft", because the term is used in most literature about open source licenses. In the following we first define the categories strong, weak and no copyleft before applying them to selected licenses. These terms are used in various literature, e.g. Schaaf (2012) and Sen et al. (2008).

Strong copyleft

A strong copyleft clause forbids the user to distribute the program and any derivates under another license. Therefore, when using a component under a strong copyleft license in a program, the whole program must be licensed under the copyleft license (Free Software Foundation, 2020d). These licenses have naturally a high legal risk because there are many restrictions imposed on the user (Synopsys Editorial Team, 2019).

Weak copyleft

This license type also includes a copyleft clause. The difference to a strong copyleft license is, that this clause does not always apply. The copyleft is limited to a specific use of the software. The original work and its derivatives in source code form are copylefted. For example, when the software is only linked to another program, and therefore in a separate file or only conveyed in object code form, the copyleft does not affect the whole program that is distributed (R. Morrison, 2018). This license category is often used for software libraries (Free Software Foundation, 2020j).

No copyleft

These licenses do not have a copyleft clause and are considered a permissive license. They permit the user to use, copy and modify the software without restrictions under the condition that the copyright and license notices are retained when distributing the software. Therefore, this license type has a low legal risk (Synopsys Editorial Team, 2019). Many of the most used licenses are now non-copyleft licenses as will be discussed more in depth in chapter 2.6.4.

A summary of the differences in how the software can be distributed and the resulting license conditions are shown in Figure 2.

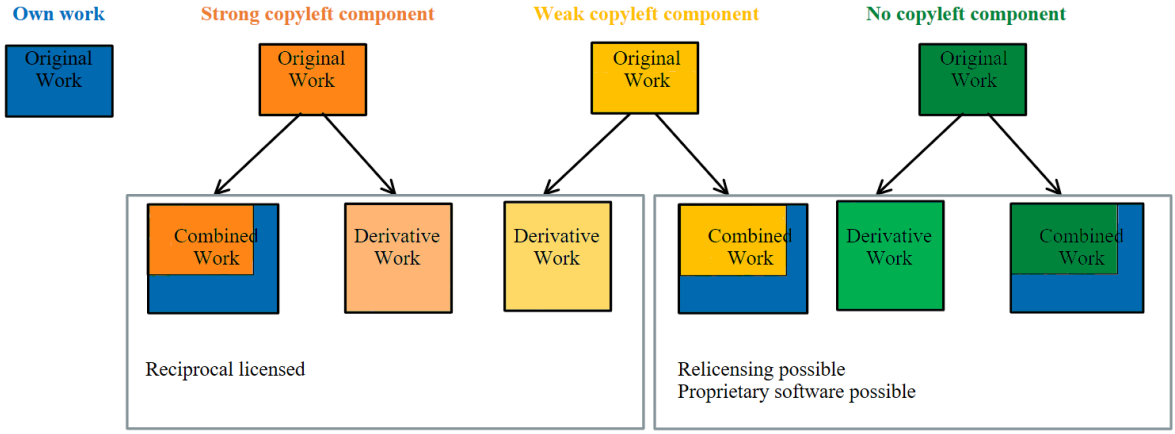

Figure 2. *Functional Differences Regarding Combination and Modification between Open Source License Categories.* Adopted from Välimäki (2005, p. 119).

OSS can be either be distributed as a derivative or a combined work. A derivative work can be created by modifying the original work. Figure 2 shows, that if there were changes made to the licensed software, distribution under another license is only authorized by no copyleft licenses. When creating a combined work, both non and weak copyleft licenses permit relicensing and making the software proprietary. A combined work is created when linking the own work with the licensed software (Free Software Foundation, 2020a). Strong copyleft licenses only permit distribution of combined work under the same license, which is called reciprocal licensed. The same terms apply to derivative works of weak and strong copylefted software.

2.5.4 The Top Ten Licenses

The ten most used licenses cover 94% (*Top Open Source Licenses | Black Duck Software*, 2020a) of all 2.5 million open source projects in the Black Duck KnowledgeBase (J. Goldstein, 2017). They are all approved by the OSI (Open Source Initiative, 2020b) and the FSF except the first Version of the Artistic License (Free Software Foundation, 2020c).

2.5.4.1 Categorization

We will name the ten most used licenses, assign them to the categories that were established in 2.5.3 and give some general information about them.

There are no standardized abbreviations for the different license names. We chose to use the abbreviations that were established by the Software Package Data Exchange (SPDX) project, founded by the Linux Foundation (*SPDX License List | Software Package Data Exchange (SPDX)*, 2020).

No copyleft

MIT License (MIT)

The MIT, which was published by the Massachusetts Institute of Technology, is a very short and permissive license. All conditions imposed consist of 20 words. The only requirement is the preservation of copyright and license notices in all copies of the Software. (Open Source Initiative, 2020a)

ISC License (ISC)

The ISC is even shorter than the MIT and was published by the Internet Systems Consortium, Inc. The appearance of license and copyright notice are also the only obligations. (Isc, 2020)

Apache License 2.0 (Apache-2.0)

In 2004 the Apache Software Foundations released the second version of the Apache License. It is a permissive license which allows free usage but requires the addition of a prominent notice when a file has been modified. Relicensing is allowed when the new license complies with the Apache-2.0. If the software includes a "NOTICE" text file, this file must be included in the distribution. The license text and all copyright, patent, trademark, and attribution notices must be retained. To every user, a patent license is granted. (The Apache Software Foundation, 2019)

BSD License 2.0, BSD-New, Modified BSD, Revised BSD (BSD-3-clause)

The BSD License 2.0 was created for the Berkeley Software Distribution (BSD) in 1999 by the University of California. It consists of three clauses. The first and the second clause state, that redistributions in source code or binary form must include the copyright and license notice. The third clause forbids the usage of the organization and it's contributors names to advertise products that use the software. (Free Software Foundation, 2020k)

The first version of the BSD license included an advertising clause, which obliges the user to include an acknowledgement that software developed by a specific organization is included. (Montague, 2007) As this obligation is very hard to fulfill when there are many different organizations using this license and consequently many acknowledgements to be added, this clause was removed when creating the second version (Free Software Foundation, 2020l).

There is also a Version with only the first two clauses which is called the FreeBSD license or BSD-2-clause (Free Software Foundation, 2020l).

Weak copyleft

GNU Lesser General Public License (LGPL-2.1)

The LGPL-2.1 was published in 1999 by the FSF. (Free Software Foundation, 2020h). It was designed for licensing software libraries and to make them more popular by permitting a wider usage, e.g. in proprietary programs. However, the FSF suggests not to use the LGPL for all libraries. For ones that have a unique competitive advantage, they suggest using a strong copyleft license, so that everyone who wants to use the library is forced to make their software free as a result of the strong copyleft (Free Software Foundation, 2020j).

The LGPL-2.1 has a weak copyleft, therefore the copyleft clause does not always apply. If the library is modified, it must be distributed under the same license, stay a software library and all changes must be marked. When the library is used unmodified, it depends on how it is combined with other software. (Free Software Foundation, 2020h) The FSF explained the different scenarios which concern all versions of the LGPL in their frequently asked questions. When the library is statically (prior to the execution of the program) imported into the program, its source must be distributed in an object format to enable the user to modify the library. Whereas when the library is only loaded dynamically (during the execution of the program) and also not conveyed by the distributor, the sources must not be made available. (Free Software Foundation, 2020a) Dynamic loading does not create a derivative work of the library and therefore the work is not affected by the copyleft (Free Software Foundation, 2020h).

GNU Lesser General Public License (LGPL-3.0)

The LGPL-3.0 includes the GPL-3.0 with the added permission to create a combined work which may be distributed under another license. A notice, that the software is used and covered by the LGPL must be added as well as the license text. The "Minimal Corresponding Source" must be conveyed. This means only the source code that is based on the library. (Free Software Foundation, 2020i)

Eclipse Public License 1 (EPL-1.0)

The Eclipse Foundation published the EPL-1.0 for their Eclipse projects. It is a weak copyleft license, because the copyleft only refers to code in source form, not in object code form. When the program is distributed in source code form, the license must not be changed. In object code, a new license can be chosen but it must comply with the EPL. The user has to be informed where to obtain the source code and that it's available under the EPL. If the software is commercially distributed, the distributor has to indemnify the other contributors in case of a legal action by the end user of the commercial software. (Eclipse Foundation, 2017)

The Eclipse Public License 2.0 (EPL-2.0) was published in 2017 (Milinkovich, 2020). Both versions of the license are similar, but the later one allows relicensing the program under the GPL-2.0 or any later version and can be used for scripting languages (Beaton, 2020).

Artistic License Perl (Artistic-1.0-Perl)

The Artistic License was created in 2000 by the Perl Foundation. There are also two Versions. In contrast to the second version, the Artistic-2.0, released in 2007, the first one is not accepted as a free software license by the FSF because it is "too vague" (Free Software Foundation, 2020c). Prior to 2007, the Perl Foundation dual licensed their projects under the Artistic License and a GNU General Public license. The concept of dual licensing is explained in chapter 2.6.6. The second, revised one includes patent protection and the permission to relicense modified versions of the program under specific conditions (Perl Foundation, 2020b).

We categorized the Artistic License into weak copyleft licenses because the distribution of modified versions is restricted. These versions must either be made available to a copyright holder of the standard version under the same license or be renamed and not contain the name of the standard version. Moreover, it must be possible to install and run the standard version simultaneously. Another possibility is making the software available to everyone who uses the version under the same or another free software license. It must be a copyleft license like the GPL and LGPL (Perl Foundation, 2020a). Furthermore, the Perl Foundation stated in their Notes to the license that relicensing under the Apache License is also possible (Perl Foundation, 2020b). Why this license was classified as a copyleft license is unclear (Jaeger & Metzger, 2016).

Microsoft Public License (MS-PL)

The Microsoft Public License or in short, the MS-PL (not to be confused with the MPL, the Mozilla Public License) was published by Microsoft. In source code form the work may only be distributed under this license. When distributing the software in object code or compiled form, the license may be chosen but it must comply with the MS-PL. It is not required to deliver the source code. The license grants all users patent use. (Open Source Initiative, 2020d).

Strong copyleft

GNU General Public License (GPL-2.0)

The GNU General Public license Version 2.0 is the most used strong copyleft license. It was published in June 1991 by the Free Software Foundation. Modifying the source code is allowed, but all changes must be labelled. When distributing the program either modified or not modified, it is required to make the whole source code available to all users. When only non-source forms are conveyed, it is obligatory to provide a written offer to make the source code available. Also, the license has to be retained. (Free Software Foundation, 2020f). Money can be charged for the GPL licensed program. However, it is forbidden to charge a disproportionately high fee for delivering the source code upon request of the user. To the FSF it is very important clarify that free software is free in terms of "free speech" and not "free beer" (Free Software Foundation, 2020o).

GNU General Public License (GPL-3.0)

The GPLv3 is the newest version of the GNU General Public License and was published in 2007 (Free Software Foundation, 2020g). The FSF revised the GPL to include a patent clause which protects users from patent infringements alleged by contributors and redistributors. Another change is the prevention of what Richard Stallman calls "tivoization". It refers to appliances that use GPL software but will not work when this software is modified. As this may be interpreted as a restriction to freedom, the GPL-3.0 prohibits this approach. (Stallman, 2014)

There is also another GPL licenses, which can be described as an ultra-strong copyleft license (R. Morrison, 2018). It is not one of the top ten because the usage is very specific, but we wanted to include it as it is a unique license. It is a network protective license and called the GNU Affero General Public License 3.0.

GNU Affero General Public License 3.0 (AGPL-3.0)

Also in 2007, only five months after the GPL-3.0 release, the FSF released the AGPL-3.0. (Free Software Foundation, 2020m) The reason for creating this license was the growing usage of software as a service (SaaS). In this case, the copyleft clause does not apply, because the software is not technically distributed, only hosted. This problem is referred to as the "SaaS loophole" (Odence, 2017). Therefore, the GPL-3.0 was modified by adding the requirement to make the source code, if modified, available to all users that are interacting with it (Free Software Foundation, 2020n). While this obligation deterred many users at first, the license is becoming more and more accepted (Meeker, 2016).

2.5.4.2 Changes over Time

While looking for open source license usage statistics, we noticed, that the percentages have changed very much over the years. To get an overview, we compiled all found data into the following figure. The usage percentage of the LGPL in version 2.1 and 3.0 are combined in some sources and are therefore shown as one line.

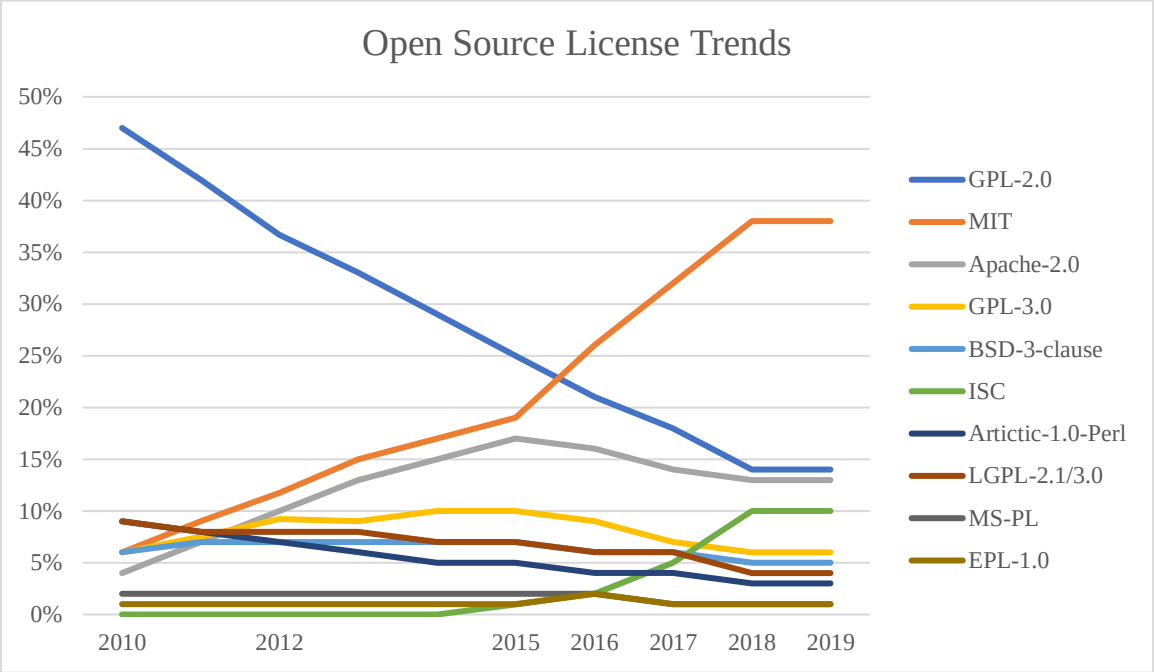

Figure 3. *Top Ten Open Source License Usage Trends from 2010 to 2019.* As the Black Duck KnowledgeBase is not publicly available anymore, the data was acquired from other literature. The Black Duck KnowledgeBase data from 2010 was extracted from an Article by Jan-Felix Schrape (2019), 2012 from the Bachelor Thesis from Alexander Schaaf (2012), 2015 from a blog entry by Scott Wilson (2015) and 2016-2019 from the Black Duck website via Wayback Machine, an initiative of the Internet Archive. (*Top Open Source Licenses | Black Duck Software*, 2020a; *Top Open Source Licenses | Black Duck Software*, 2020d; *Top Open Source Licenses | Black Duck Software*, 2020c; *Top Open Source Licenses | Black Duck Software*, 2020b)

The top ten licenses stayed the same in the last 10 years, but the percentage of their usage has changed a lot. Figure 3 indicates that there is a clear rise of the usage of the MIT license and a drastic decline of the GPL-2.0 usage. While in 2010 the GPL-2.0 was very popular, the analysis of Donnie Berkholz (2013) showed, by analyzing 17,549 open source projects, a trend from copyleft to permissive licenses. Stephen O'Grady (2012) explains this development with the argument that software is more and more used as a service and the value of written software declines as it is no more a competitive advantage. Therefore, he says that choosing a permissive license makes sense due to the benefits of making source code available.

The figure shows, that the ISC and the Apache-2.0 are also getting used more and more over the years. In the following figure we divided all ten licenses into copyleft and non-copyleft to see if there was a general trend to more permissive licenses from 2010 onwards.

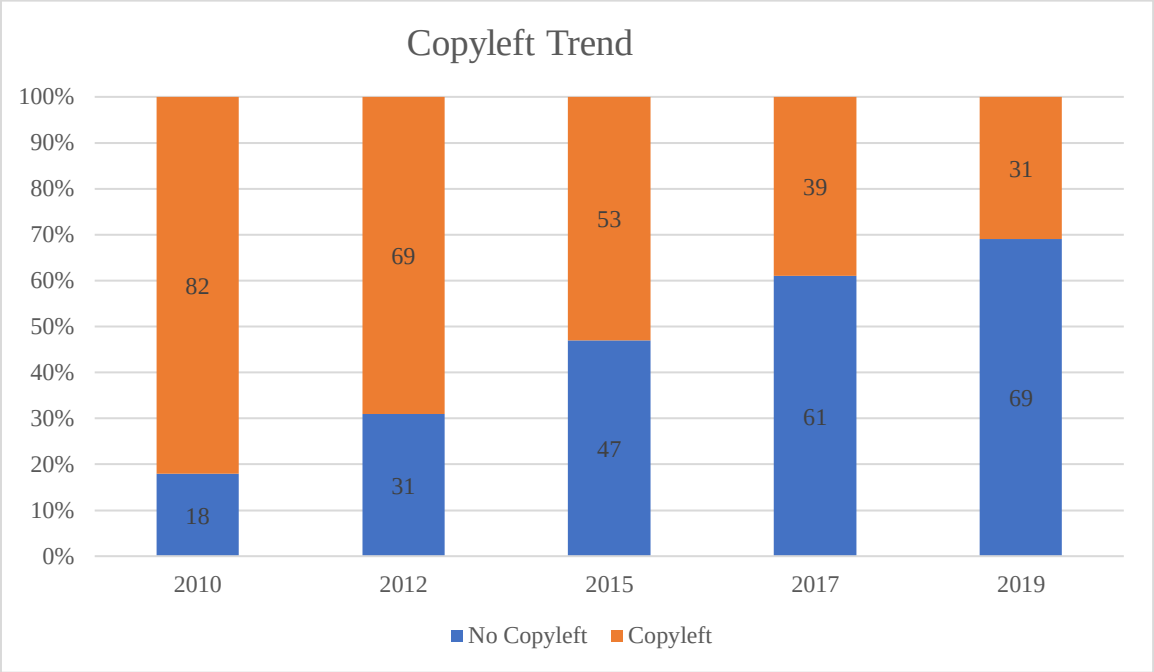

Figure 4. *Trend of No Copyleft and Copyleft Licenses from 2010 to 2019.* As the Black Duck KnowledgeBase is not publicly available anymore, the data was acquired from other literature. The Black Duck KnowledgeBase data from 2010 was extracted from an Article by Jan-Felix Schrape (2019), 2012 from the Bachelor Thesis from Alexander Schaaf (2012), 2015 from a blog entry by Scott Wilson (2015) and 2016-2019 from the Black Duck website via Wayback Machine, an initiative of the Internet Archive. (*Top Open Source Licenses | Black Duck Software*, 2020a; *Top Open Source Licenses | Black Duck Software*, 2020d; *Top Open Source Licenses | Black Duck Software*, 2020c; *Top Open Source Licenses | Black Duck Software*, 2020b)

Figure 4 indicates a clear trend towards non-copyleft licenses.

A. Goldstein (2020) explains the rise of the incline of non-copyleft licenses with the general increase of open source software usage. Even commercial software developers are now supporting open source software. Permissive licenses are preferred because compliance is much easier.

Commercial software companies also a reason, why copyleft licenses like the GPL-2.0 are in decline. Their ideology of making software free is rarely compatible with commercial business models says Bacon (2017).

Balter, an attorney working at GitHub, explained in an Interview, why specifically the MIT gained so much popularity. According to him, the license is ideal for developers because it is short and easy to understand. (Todorović, 2020)

The MIT is not only the most used license in the Black Duck KnowledgeBase. It is also number one on GitHub (Balter, 2015) and in the WhiteSource database which consist of 130 million open source files (A. Goldstein, 2020).

These developments do not mean, that the strong copyleft licenses are going to disappear. Berg, an open-source licensing consultant explained to the Register that they are used increasingly in cloud computing. Especially the AGPL is used because the developers want the user to be a member of the community and not only use the software to solely benefit themselves. (Claburn, 2020)

Both strong and non-copyleft licenses have their advantages. It is not possible to say which one is freer. G. Morrison (2015) concludes that permissive licenses like the BSD give more freedom

to the developers. End users benefit more from restrictive licenses like the GPL because it grants them more freedom.

2.5.5 License Comparison

To compare the licenses we used the categories permissions, conditions and limitations, based on the choosealicense.com website (*Appendix*, 2020). We analyzed the license texts and looked for phrases that identify the categories' prominent features. The corresponding paragraphs were added to an excel sheet. We noticed, that there are general terms that are covered by most licenses. Those are the following:

Permissions

- Distribution under another license
- Proprietary use
- Patent rights grant from contributors
- Private use
- Commercial use

Conditions

- Source disclosure
- Include license and copyright notice
- State changes
- Special obligations

Limitations

- Limitation of liability
- No provision of warranty

To abstract the collected information, we created a comparison table. In the comparison table we did not include the common features. Private and commercial use are permitted by all licenses. The license and copyright notices must always be retained. Every license includes limitations of Liability and Warranty. For special obligations we added an extra column.

We also found, that copyleft licenses in general impose more obligations on the user. They not only restrict the license choice by a copyleft clause, the user is also often obliged to state changes made to the source code and to disclose the source where the original source code can be obtained.

Table 1

Comparison of the Top Ten Open Source Licenses

	Permissions			**Conditions**		
	Distribution under another license	**Proprietary use**	**Patent grant from contributors**	**Source disclosure**	**State changes**	**Special Obligation**
MIT	Permitted	Permitted	No	No	No	No
ISC	Permitted	Permitted	No	No	No	No
BSD-3-clause	Permitted	Permitted	No	No	No	Promotion with author name is forbidden without prior permission
Apache-2.0	Permitted	Permitted	Yes	No	Yes	Notice file must be included
EPL-1.0	Restricted	Restricted	Yes	Yes	No	Compensate contributors for damages caused by commercial offering
MS-PL	Restricted	Restricted	Yes	No	No	No
Artistic-1.0-Perl	Restricted	Restricted	Yes	Yes	Yes	Modified versions must be renamed and made available
LGPL-2.1	Forbidden	Restricted	No	Yes	Yes	May only be used as a library with proprietary code
LGPL-3.0	Forbidden	Restricted	Yes	Yes	Yes	May only be used as a library with proprietary code
GPL-2.0	Forbidden	Forbidden	No	Yes	Yes	Give access to complete source code
GPL-3.0	Forbidden	Forbidden	Yes	Yes	Yes	Give access to complete source code
AGPL-3.0	Forbidden	Forbidden	Yes	Yes	Yes	Network use of modified versions is copylefted

2.5.6 Dual Licensing

OSS is not only developed by private persons or specific open source communities. There are also many companies that develop open source software. Oracle for example makes MySQL, "the world's most popular open source database" (*MySQL: Commercial License for OEMs, ISVs and VARs*, 2020) available under a commercial license for distributors of commercial applications, but also under the GPL for distributors of open source applications. They even added

an exception to the GPL, so that not the entire derivative work is affected by the copyleft in the GPL (*MySQL: Commercial License for OEMs, ISVs and VARs*, 2020).

Dual licensing with a GPL license and a commercial one is the most popular way to use multiple licenses (Jacobs, 2017).

It is also possible to license a component under multiple open source licenses from which the user then can select the one that fits his project best. Often there is the possibility to choose between a strong copyleft and more permissive license. For example, the programming language Perl is licensed under the Artistic-1.0-Perl and the GPL-1.0 or any later version. (*Perl Licensing - Dev.Perl.Org*, 2020)

2.5.7 License Compatibility

When developing a software product, it often depends on components or other projects from third parties. These third-party components can either be closed or open source and are interdependent. The used open source components can all have different licenses, which can lead to license incompatibilities. (Riehle & Harutyunyan, 2019)

The copyleft clause often causes these incompatibilities. If there are one or more copylefted components included, Jaeger and Metzger (2016) suggest to answer the following questions:

1. Are two or more software components creating a derivative work as defined by the applying license?
2. Are copyleft licenses affected?
3. Do Non-Copyleft licenses contain obligations, that the copyleft licenses do not?

If the first question can be answered with “no”, no incompatibilities are caused. When the different programs are only use side by side and are not combined into one program, there is no copyleft-effect and therefore no incompatibility. If the first question is affirmed as well as the second one, the licenses can only be compatible if there is a compatibility clause like it is in the Artistic License. If there is no such clause, the components are incompatible because copyleft license “A” demands to license the whole work under “A” and “B” to be licensed under “B” which is impossible to achieve. (Jaeger & Metzger, 2016)

Usually non-copyleft licenses are compatible with other licenses as they do not have many restrictions. But is also possible, that a non-copyleft license causes incompatibilities. Because of that, the third question must be answered. An example is the advertising clause of the first BSD license. The GPL-2.0 does not have a clause like that but one that forbids to impose any further restrictions, so therefore the two licenses are not compatible. (Jaeger & Metzger, 2016)

The GNU (Lesser) General Public Licenses can either be licensed under a single version or include the permission to update the license to any later version. This option has a high impact on the compatibility. Richard Stallman, the founder of the FSF urges developers to license their work under GPL and any version later to avoid upcoming incompatibilities (Stallman, 2018) We will visualize this option by a “+” in the abbreviation.

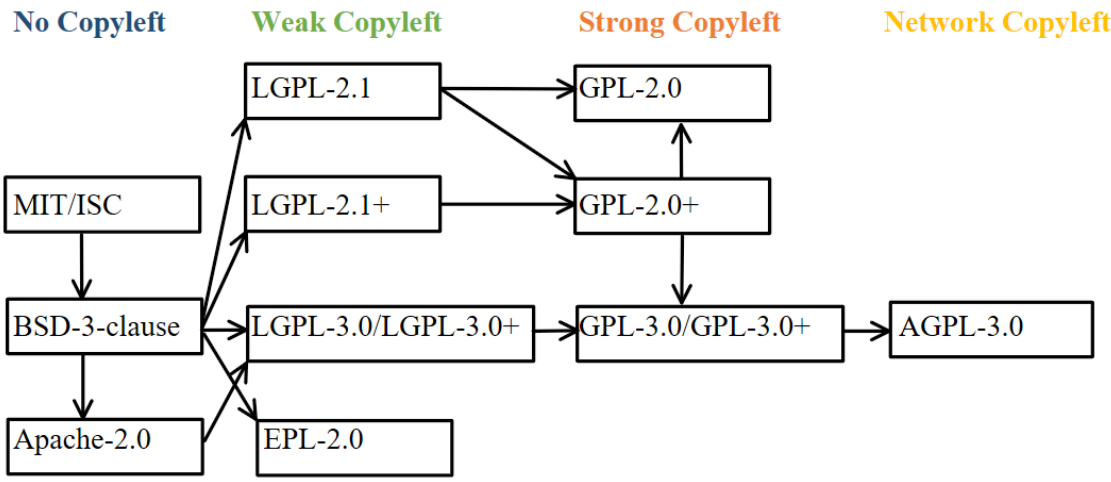

Figure 5. *Directed graph showing the compatibility of open source licenses.* Adapted from Wheeler (2017). The EPL-2.0 was added according to Morrison (2018) Two licenses are compatible if both can reach the same license by following the arrows or one can reach the other. The combined software must then be relicensed to the reached license.

The license compatibilities of some of the top ten licenses are shown in Figure 5. When an arrow is drawn from one license (A) to another (B), the two licenses are compatible. The compatibility only works in one direction. When the software is combined, the resulting work must be licensed under B. If there is an arrow from B to another license (C), the work can also be licensed under C and all other licenses on this path. Two different licenses can be combined when both can reach a common license, or one can reach the other.
For example, LGPL-2.1 and Apache-2.0 cannot be directly connected by one or more arrows, but both can reach GPL-3.0 or GPL-3.0+. Therefore, they can be combined under the GPL-3.0 or GPL-3.0+.

Due to incompatibility, some of the top ten licenses are not included in Figure 5. As the Artistic-1.0-Perl is not approved by the FSF, it is not compatible with any GNU licenses. The Artistic-2.0 however is compatible with the GPL-2.0 and GPL-2.0+ because of the added relicensing option. The MS-PL is not compatible with any GNU licenses. (Free Software Foundation, 2020c)

The compatibilities of the GNU licenses can change depending on how the software is used. When the code is embedded in another software, Table 2.1 can be used to identify compatibilities. When the software is used as a library Table 2.2 can be used.

Table 2.1

Compatibility of GNU Licenses when code is intended to be embedded

		Code to be licensed under:					
		GPL-2.0	**GPL-2.0+**	**GPL-3.0+**	**LGPL-2.1**	**LGPL-2.1+**	**LGPL-3.0+**
Code to be used is under:	**GPL-2.0**	OK	OK: Convey copied code under GPL-2.0	NO	OK: Combination is under GPL-2.0	OK: Combination is under GPL-2.0	NO
	GPL-2.0+	OK: Combination is under GPL-2.0	OK	OK	OK: Combination is under GPL-2.0+	OK: Combination is under GPL-2.0+	OK: Combination is under GPL-3.0
	GPL-3.0	NO	OK: Combination is under GPL-3.0	OK	OK: Combination is under GPL-3.0	OK: Combination is under GPL-3.0	OK: Combination is under GPL-3.0
	LGPL-2.1	OK: Convey copied code under GPL-2.0	OK: Convey copied code under GPL-2.0+	OK: Convey copied code under GPL-3.0+	OK	OK: Convey code under LGPL-2.1	OK: Convey copied code under GPL-3.0
	LGPL-2.1+	OK: Convey copied code under GPL-2.0	OK: Convey copied code under GPL-2.0+	OK: Convey code under GPL-3.0+	OK: Combination is under LGPL-2.1	OK	OK
	LGPL-3.0	NO	OK: Combination is under GPL-3.0	OK: Combination is under GPL-3.0	OK: Combination is under GPL-3.0	OK: Combination is under LGPL-3.0	OK: Combination is under LGPL-3.0

Note: Based on the Free Software Foundation (2020b). If code, licensed under one of the licenses in the left column, is to be embedded in software which the developer wants to license under one of the licenses in the top row, the cell both licenses have in common tells if this action is possible. “OK”: Action is possible with or without further modifications of the resulting license. “NO”: Action is not possible. “NO” is highlighted red to better distinguish it from “OK”.

Table 2.2

Compatibility of GNU licenses when code is intended to be used as a library

		I want to license my code under:					
		GPL-2.0	**GPL-2.0+**	**GPL-3.0+**	**LGPL-2.1**	**LGPL-2.1+**	**LGPL-3.0+**
I want to use a library under:	**GPL-2.0**	OK	OK	NO	OK: Combination is under GPL-2.0	OK: Combination is under GPL-2.0	NO
	GPL-2.0+	OK	OK	OK	OK: Combination is under GPL-2.0+	OK: Combination is under GPL-2.0+	OK: Combination is under GPL-3.0
	GPL-3.0	NO	OK: Combination is under GPL-3.0	OK	OK: Combination is under GPL-3.0	OK: Combination is under GPL-3.0	OK: Combination is under GPL-3.0
	LGPL-2.1	OK	OK	OK	OK	OK	OK
	LGPL-2.1+	OK	OK	OK	OK	OK	OK
	LGPL-3.0	NO	OK: Combination is under GPL-3.0	OK	OK	OK	OK

Note: Based on the Free Software Foundation (2020b). If code, licensed under one of the licenses in the left column, is to be used as a library in software which the developer wants to license under one of the licenses mentioned in the top row, the cell both licenses have in common tells if this action is possible. “OK”: Action is possible with or without further modifications of the resulting license. “NO”: Action is not possible. “NO” is highlighted red to better distinguish it from “OK”.

According to Table 2.1 and 2.2, Code under a GPL license can be used as a library with LGPL code if the combination is going to be licensed under the corresponding version of the GPL. Only the GPL-2.0 an the LGPL-3.0+ cannot be combined. Code under the LGPL-2.1 can be relicensed to GPL-2.0 and any later version. When used as a library, the resulting work must not be relicensed. It is not allowed to relicense code under a lower version to a newer one if the older one does not have the “any version later” option. Every other combination is possible, but in some cases, the resulting work must be relicensed. (Free Software Foundation, 2020b)

2.5.8 Which License to Choose

It can be very difficult to choose the right license for an open source project. Sen et al. (2008) found out, that it is especially very difficult for developers. One reason is, that there is not much literature about choosing a license from a developer’s perspective. Another reason are the conflicting motivations and attitudes of developers. On the one hand, they want to choose a copyleft license, so the software stays open source, on the other hand, they do not want to restrict the redistribution. (Sen et al., 2008)

In the following, we will first establish a decision tree in Figure 6, which helps to find the right license strategy. We will then provide decision trees in Figures 7.1, 7.2 and 7.3 to make a specific license decision based on the before selected license category.

2.5.8.1 Choice of License Strategy

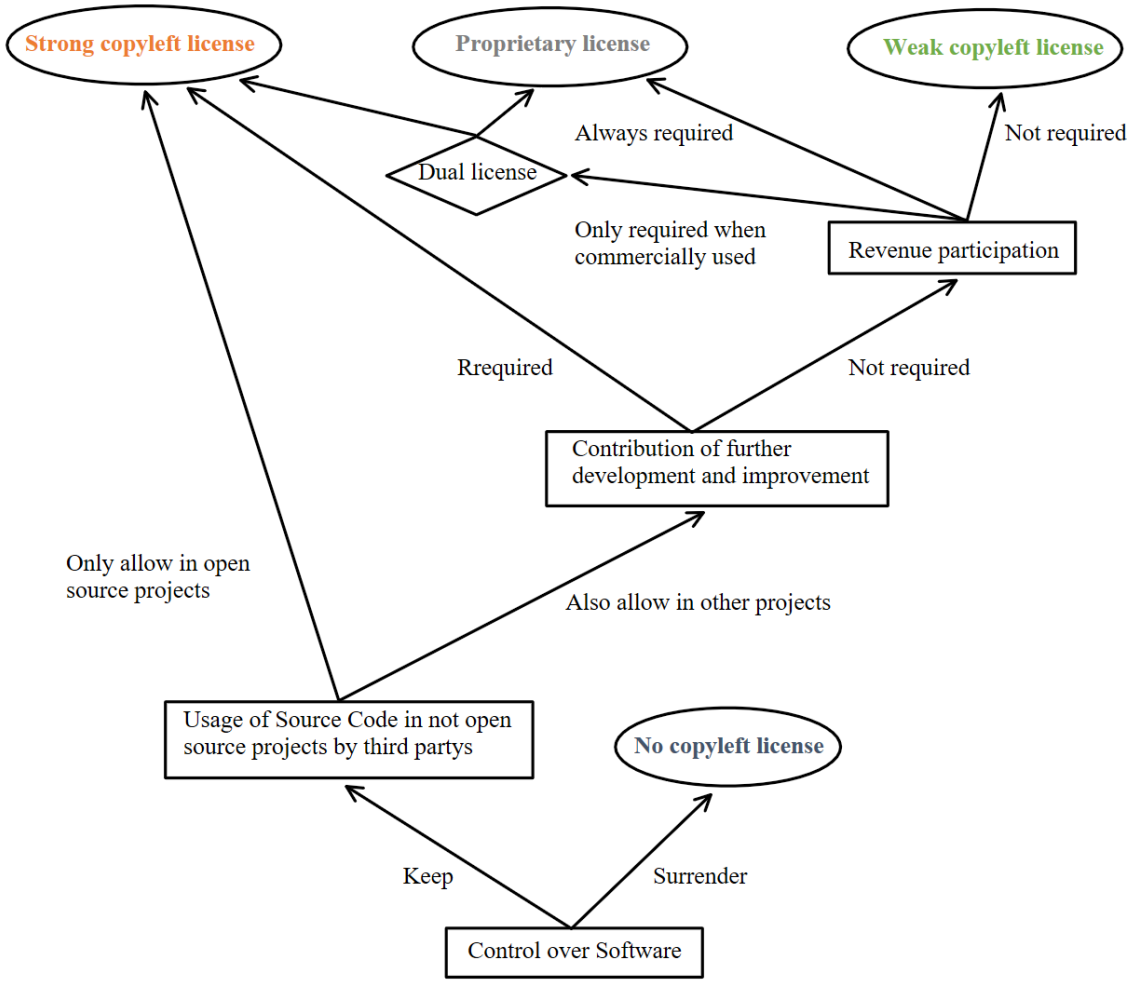

Figure 6. Decision tree for choosing a license strategy modified after (Alexander Schaaf et al., 2013, p. 168)

To choose a strategy type, several questions must be answered, which are shown in Figure 6. The first one is, if the distributor wants to have control over the software after releasing it. If not, a license without a copyleft license should be chosen. The user can then use the software without restrictions except for liability and warranty.

In case the licensor only wants to allow the use of the software in other open source environments, a strong copyleft license must be chosen. This license category should also be chosen if the contribution of further development and improvements is required.

If as well usage in only open source projects as contribution is required, it must be decided if revenue participation is also required. If so, a proprietary license must be chosen. Otherwise, a weak copyleft license meets all requirements. It is also possible to differentiate when revenue participation is required. In this case, the software can be dual licensed with a proprietary and a strong copyleft license. The user must then choose the proprietary one when the software is commercially used.

2.5.8.2 Choice of OSS License

After the decision has been made which license category is most suitable for the product, a specific license must be selected. We therefore developed decision trees to help making this

choice. We orientated ourselves towards the data of Table 1. We examined what sets the licenses of the different categories apart from each other.

If other OSS is included, compatibilities must be considered. The permission to update the version of a GLP or LGPL license can always be added and is an important licensing choice.

When the no copyleft license category has been chosen and the licensor does not want to impose any further restrictions to the user, the MIT or ISC license should be chosen. The common features of all licenses are the condition to retain license and copyright notices and the limitations of liability and warranty, as discussed in chapter 2.5.5. To forbid the use of the name of the copyright holder or the contributors for promotion purposes, the BSD-3-clause is suitable. The Apache-2.0 license should be chosen if the user should be granted a patent license.

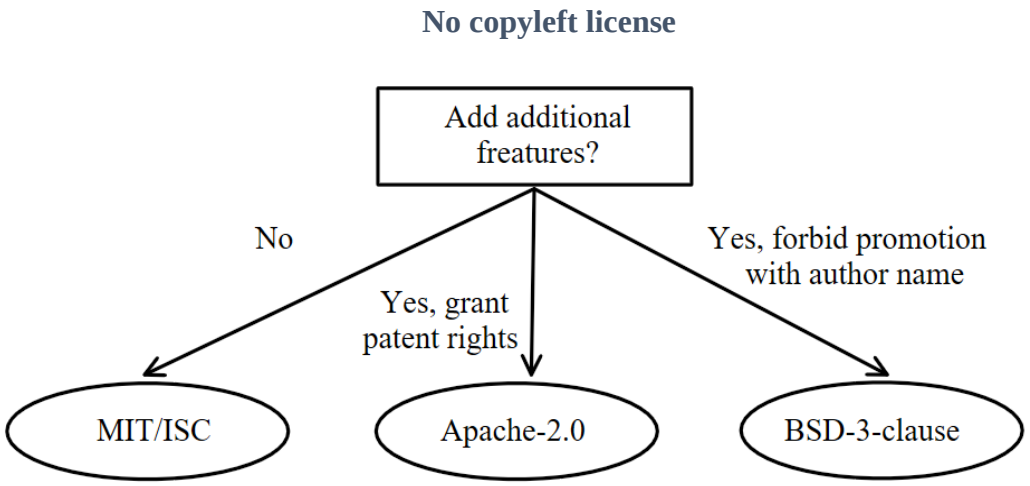

Figure 7.1. Choice of a no copyleft license

For deciding on a specific weak copyleft licenses, the following choices, shown in Figure 7.2, should be considered. If modified versions of the software should be copylefted, the Artistic-License-1.0-Perl is most suitable. Both the MS-PL and the EPL-1.0 only impose the copyleft clause on software in source code form. If also a grant of patent rights should be included, the MS-PL should be chosen. The LGPL-3.0 also includes a clause that grants the user patent rights. This license is suitable for software libraries and the copyleft should not apply when the software is only linked. In this case, the prior version, the LGPL-2.1 is also suitable, but it does not include a grant of patent rights.

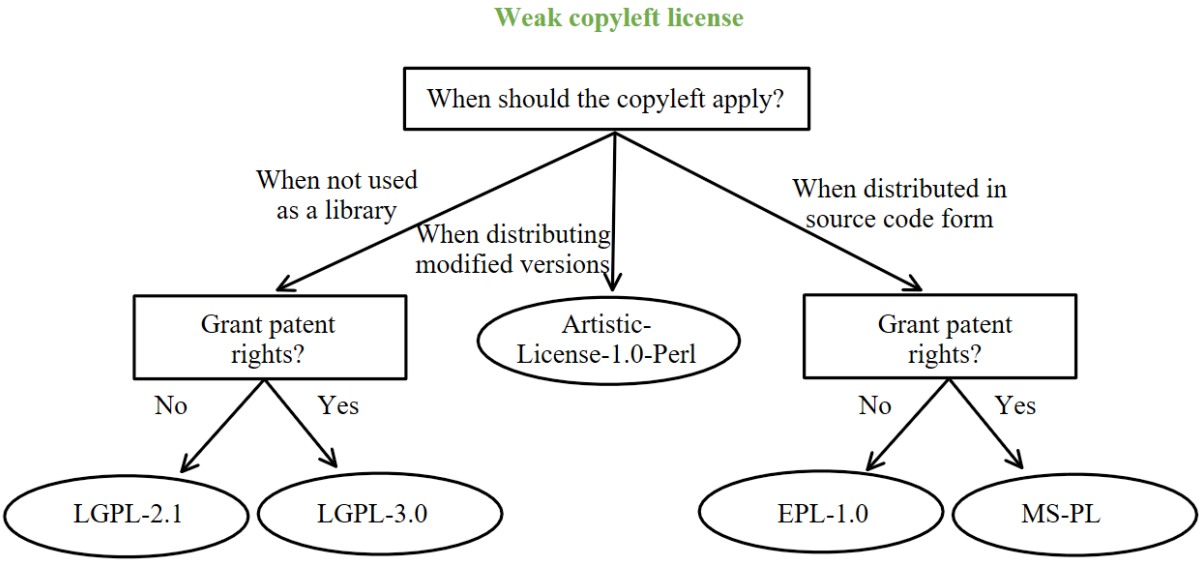

Figure 7.2. Choice of a weak copyleft license

Concerning strong copyleft licenses, it must be considered whether the use of the software on a network should be copylefted. If network use should be considered a distribution, the AGPL-3.0 must be chosen. If, however, network use is irrelevant or should not be copylefted, the GPL should be chosen. The GPL-3.0 if patent rights should be granted, the GPL-2.0 if not.

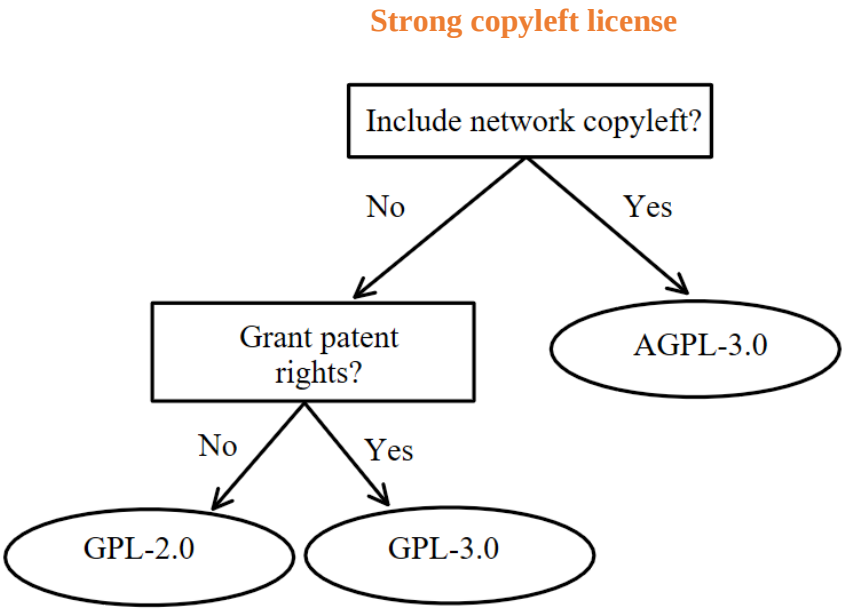

Figure 7.3. Choice of a strong copyleft license

2.6 Results Discussion

We found that there are two main organizations which deal with free and open source licenses and have also developed an approval process for the licenses: the FSF and the OSI. Both have defined certain conditions that must be met by the license in order to be approved. The definitions are similar but not identical. There are some licenses that are not approved by both organizations.
We think that these approval processes are very important as they give licensee and licensor a certain security that the basic permissions (use, copy, modify, distribute) are granted.

Our research shows that when defining different types of licenses, the differentiation between no, weak and strong copyleft is very common. Licenses can either contain a copyleft clause, a weakened copyleft clause or no copyleft clause at all. This is a key characteristic of a license because it tells the licensee whether it is allowed to distribute the software under another license. Furthermore, copyleft licenses in general impose more conditions on the user while no copyleft licenses give more unrestricted permissions. Therefore, no copyleft licenses are often called "permissive" and copyleft licenses "restrictive".
Using this categorization, a user can easily identify, whether a specific software could be of high or low legal risk. It is also important to notice, that many restrictions like the copyleft clause are not created to restrict any freedom. They ensure the freedom of the end user to receive the source code, sometimes even of the complete work. In this way, the software stays open source and can be further develop by everybody using it.

The search after the ten most used licenses has led to a surprising result. Over ninety percent of open source software (in the Black Duck knowledge base) is licensed under one of ten licenses. This has been the case since at least ten years, but the order of rank has changed over time.
A possible reason for this low diversity could be that all top ten licenses are universally applicable. They are written to be easily applied and reused. It also makes sense to reuse an already existing and commonly used license because it reduces the developers time and effort spent on formulating own terms. Choosing from a limited pool of licenses is also easier for users because that way they do not have to know many different licenses.
A general trend to non-copyleft licenses was identified. One possible explanation is the increasing use of OSS in companies, who develop proprietary software. For this purpose, strong copylefted code cannot be used. Non-copyleft licenses are also in general shorter and easier to understand.

When examining the different licenses, at the first look, they are very different. Some license texts can be very complex and hard to understand, whereas others are very short and terse. Some grant the user nearly complete freedom in what to do with the software, others impose a lot of obligations. They were therefore compared based on their permissions, conditions, and limitations. This differentiation is not very common in literature but helps to differentiate the licenses from each other. A closer look shows that there are actually many similarities between the licenses. Especially the limitations are the same for all ten different licenses. There are also some permissions that are granted by all licenses.
This additional categorization helps developers to make a license choice for own software. It also helps practitioners to decide whether a program can be used in their project. Any other license can be compared and examined based on these characteristics.

We found out, that besides the characteristics of the licenses mentioned above, there are other aspects that must be considered when using open source software. Overall, license compliance must be achieved. This means that all obligations imposed must be fulfilled. It is possible that a work is licensed under multiple licenses, so that the user must make a choice. When open

source software with different licenses are used in one program, their licenses must be compatible. Some licenses can be relicensed, to ensure compatibility. When relicensing is done, all obligations and copyright notices must be retained. To achieve license compliance, it is not sufficient to only know the specific license text and interpret and comply with them.
Based on the knowledge obtained from answering the previous research questions, a license choice can be made with the help of the established decision trees. We recommend to firstly decide on a license strategy and secondly on a specific license. Due to the comparison table developed in this thesis, it is easy to determine which decisions must be made.

2.7 Limitations

There are several limitations in this thesis caused by different aspects.

Due to the approach

Firstly, the sample study approach does have inherent limitations. It could have been possible, that all top ten used licenses are very similar and belong to the same category. Luckily, the top ten licenses cover a large variety of permissions and conditions. But it could be possible that there are licenses which contain additional aspects that are not represented in the top ten. It could be, that there are licenses that cannot be fitted into these categories, even if the categorization tries to include all possible licenses.

Secondly, a systematic literature review can miss relevant sources. Only sources in English and German were reviewed. We can therefore not know if there are relevant works in other languages. Through the application of exclusion criteria, relevant sources could have been excluded. Additionally, even though a systematic literature review is an unbiased method, the sources can be biased.

Due to narrowed focus

This research is focused on the technical aspects and usage. Juridical aspects such as judgements were not examined.

Due to changes in the field

Open source licensing has only existed for 30 years and has only become well known in the last years. Therefore, there is not very much literature referring to this topic. The matter is also ever changing. Technical innovations can make new license obligations necessary, like it was with the SaaS loophole. License preferences change as discussed in chapter 2.5.4.2.

2.8 Future Work

The introduced model must undergo a user test. Its practicality must be examined. To verify its legal accuracy a lawyer should examine it.

Even though the ten most used licenses cover a lot of projects, there are many more that use their own or a more unpopular license. In different communities, different licenses are popular and broadly used. These licenses must be examined too. As they change over time It is important to look at the license developments regularly and look out for new licenses that are becoming popular.

It would also be interesting to examine if specific licenses are preferred for certain applications or based on other reasons, like age and origin of the developer.

A survey to find out why developers often do not add a license could be conducted. It would be interesting to know if they simply do not know about licensing, do not want to spend time on learning about the different licenses or if they intentionally do not add one and if so why.

An automated process which scans the source code for license information (open source scanner tools are already available, e.g. Fossology) and then lists the different permissions, conditions and limitations of the licenses included could be developed. Through this grouping, the obligations of many licenses can be condensed. The compatibilities should also be included. To make this possible, many more licenses must be examined, and a comprehensive compatibility matrix must be created.

2.9 Conclusion

This thesis aimed to create a model of open source licenses. Due to the large number of different licenses and time restrictions a sample study approach was applied. The ten most frequently used licenses were analyzed. They cover a large amount of open source software. A systematic literature review was carried out to gather all relevant information.

The results show that there are two main organizations that have defined rules about what an open source license should grant. Most common licenses follow these rules and are therefore approved by these organizations.

We found that there are three different types of open source licenses. Furthermore, licenses can be compared based on the permissions, conditions, and limitations they impose on the user. By defining these characteristics, the model helps researchers to classify new licenses and to compare them to commonly used ones. As the model was only applied to ten licenses, there are possible restrictions. However, other licenses usually do not include new aspects but contain the discussed ones, only differently arranged and phrased (Jaeger & Metzger, 2016, p. 26). Nevertheless, sometimes new technology demands adjustments to licenses or even new licenses. Therefore, more licenses must be examined in future work.

Besides the specific contents of the license text, important principles like license compatibility, compliance and dual licensing were analyzed.

Practitioners are given a framework to firstly choose a licensing strategy and secondly a specific open source license based on the differences of the licenses. It is not legal advice but helps to understand the basic concept of open source licensing and which decisions should be considered.

Appendix License Implementation

When distributing a work under an OSS license, a license notice must always be added. In case of a non-copyleft licenses, a text file including the license text at the root directory is sufficient. (R. Morrison, 2018)

For the Apache-2.0 it is suggested to include a copy of the license text in a file called “LICENSE”. If only single files are to be licensed, the following text should be added including the authors copyright information in the . (The Apache Software Foundation, 2019)

```
Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

(The Apache Software Foundation, 2019)

Weak and strong copyleft licenses usually require more effort. For applying any GNU license, each source file should contain a copyright disclaimer and a license notice. Information on the authors contact information should be included. Furthermore, a file named “COPYING” should be added, containing a copy of the corresponding license text. If the LGPL is used, a “COPYING.LESSER” file should be added. In case of the GPL-3.0+, the following license notice is suggested. The license version can be replaced with any other version of the GPL. (Free Software Foundation, 2020p)

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

(Open Source Initiative, 2020e)

To make license declarations easier, the Free Software Foundation Europe (FSFE) developed the REUSE project. A project is REUSE compliant if every file includes a copyright and licensing information. The FSFE provides different tools, to achieve this. It not only makes licensing easier for distributors. Because the license and copyright information is added in a standardized way, it makes it also much easier for other tools like license scanner to seek out the needed information. (Free Software Foundation Europe, 2020)

GitHub also provides developers an easy licensing process. To add a license to a repository, a new file named “LICENSE” or “LICENSE.md” must be created and then, a license template can be chosen. Most of the licenses that were discussed in this research are included. (GitHub, 2020)

References

The Apache Software Foundation. (2019, December 25). *Apache License, Version 2.0*. https://www.apache.org/licenses/LICENSE-2.0

Appendix. (2020, February 29). https://choosealicense.com/appendix/

Bacon, J. (2017). *The decline of GPL?* https://opensource.com/article/17/2/decline-gpl

Balter, B. (2015). *Open source license usage on GitHub.com - The GitHub Blog*. https://github.blog/2015-03-09-open-source-license-usage-on-github-com/

Beaton, W. (2020, March 23). *EPL-2.0 FAQ | The Eclipse Foundation*. https://www.eclipse.org/legal/epl-2.0/faq.php#h.a0eux401qus

Berkholz, D. (2013). *Quantifying the shift toward permissive licensing*. https://redmonk.com/dberkholz/2013/04/02/quantifying-the-shift-toward-permissive-licensing/

Brereton, P., Kitchenham, B., Budgen, D., Turner, M., & Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, *80*(4), 571–583. https://doi.org/10.1016/j.jss.2006.07.009

Build software better, together. (2020, April 8). https://github.com/

Claburn, T. (2020, January 17). Copy-left behind: Permissive MIT, Apache open-source licenses on the up as developers snub GNU's GPL. *The Register*. https://www.theregister.co.uk/2020/01/17/mit_apache_versus_gpl/

*Do What The F*ck You Want To Public License.* (2020, May 18). https://choosealicense.com/licenses/wtfpl/

Eclipse Foundation. (2017, November 6). *Eclipse Public License - Version 1.0*. https://www.eclipse.org/legal/epl-v10.html

Free Software Foundation. (2020a, March 3). *Frequently Asked Questions about the GNU Licenses - GNU Project - Free Software Foundation*. https://www.gnu.org/licenses/gpl-faq#LGPLStaticVsDynamic

Free Software Foundation. (2020b, March 3). *Frequently Asked Questions about the GNU Licenses - GNU Project - Free Software Foundation*. https://www.gnu.org/licenses/gpl-faq#AllCompatibility

Free Software Foundation. (2020c, March 10). *Various Licenses and Comments about Them - GNU Project - Free Software Foundation*. https://www.gnu.org/licenses/license-list.en.html

Free Software Foundation. (2020d, March 10). *What is Copyleft? - GNU Project - Free Software Foundation*. https://www.gnu.org/licenses/copyleft.en.html

Free Software Foundation. (2020e, March 10). *What is free software? - GNU Project - Free Software Foundation*. https://www.gnu.org/philosophy/free-sw.en.html

Free Software Foundation. (2020f, March 11). *GNU General Public License v2.0 - GNU Project - Free Software Foundation (FSF)*. https://www.gnu.org/licenses/old-licenses/gpl-2.0-standalone.html

Free Software Foundation. (2020g, March 11). *GNU General Public License v3.0 - GNU Project - Free Software Foundation (FSF)*. https://www.gnu.org/licenses/gpl-3.0-standalone.html

Free Software Foundation. (2020h, March 11). *GNU Lesser General Public License v2.1 - GNU Project - Free Software Foundation (FSF)*. https://www.gnu.org/licenses/old-licenses/lgpl-2.1-standalone.html

Free Software Foundation. (2020i, March 23). *GNU Lesser General Public License v3.0 - GNU Project - Free Software Foundation (FSF)*. https://www.gnu.org/licenses/lgpl-3.0-standalone.html

Free Software Foundation. (2020j, March 23). *Why you shouldn't use the Lesser GPL for your next library - GNU Project - Free Software Foundation*. https://www.gnu.org/licenses/why-not-lgpl.en.html

Free Software Foundation. (2020k, March 27). *License:BSD-3-Clause - Free Software Directory*. https://directory.fsf.org/wiki/License:BSD-3-Clause

Free Software Foundation. (2020l, March 28). *BSD License Problem - GNU Project - Free Software Foundation*. https://www.gnu.org/licenses/bsd.en.html

Free Software Foundation. (2020m, April 1). *GNU Affero General Public License - GNU Project - Free Software Foundation*. https://www.gnu.org/licenses/agpl-3.0.en.html#content

Free Software Foundation. (2020n, April 1). *Why the GNU Affero GPL - GNU Project - Free Software Foundation*. https://www.gnu.org/licenses/why-affero-gpl.en.html

Free Software Foundation. (2020o, April 15). *Selling Free Software - GNU Project - Free Software Foundation*. https://www.gnu.org/philosophy/selling.en.html

Free Software Foundation. (2020p, May 5). *How to use GNU licenses for your own software - GNU Project - Free Software Foundation*. https://www.gnu.org/licenses/gpl-howto.en.html

Free Software Foundation. (2020q, May 12). *GNU General Public License v1.0 - GNU Project - Free Software Foundation (FSF)*. https://www.gnu.org/licenses/old-licenses/gpl-1.0-standalone.html

Free Software Foundation Europe. (2020, May 6). *REUSE - Make licensing easy for everyone*. https://reuse.software/

Gentemann, L., & Termer, F. (2020). *Open Source Monitor 2019*. https://www.bitkom.org/Bitkom/Publikationen/Open-Source-Monitor-2019

GitHub. (2020, June 8). *Adding a license to a repository - GitHub Help*. https://help.github.com/en/github/building-a-strong-community/adding-a-license-to-a-repository

Goldstein, A. (2019). *Open Source Licenses Explained: Open Source Licenses Explained*. https://resources.whitesourcesoftware.com/blog-whitesource/open-source-licenses-explained

Goldstein, A. (2020). *Open Source Licenses in 2020: Trends and Predictions*. https://resources.whitesourcesoftware.com/blog-whitesource/top-open-source-licenses-trends-and-predictions

Goldstein, J. (2017, May 17). Black Duck Software – Managing and Securing Your Open Source Software. *VpnMentor*. https://www.vpnmentor.com/blog/black-duck-software-managing-securing-open-source-software/

Isc. (2020, March 25). *Internet Systems Consortium*. https://www.isc.org/licenses/

Jacobs, M. (2017, February 24). What is dual licensing? 3 software licensing models to consider. *Synopsys*. https://www.synopsys.com/blogs/software-security/software-licensing-decisions-consider-dual-licensing/

Jaeger, T., & Metzger, A. (2016). *Open Source Software: Rechtliche Rahmenbedingungen der Freien Software* (4. Auflage). C.H. Beck. https://beck-online.beck.de/Bcid/Y-400-W-JaegerMetzgerHdbOSS

Jane Webster, & Richard T. Watson (2002). Analyzing the Past to Prepare for the Future: Writing a Literature Review. *Undefined*. https://pdfs.semanticscholar.org/2d1c/e5f4b8d57fa659ed7e49a50531180f0e0fef.pdf

Kitchenham, B., & Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering, *2*.

Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering – A systematic literature review. *Information and Software Technology*, *51*(1), 7–15. https://doi.org/10.1016/j.infsof.2008.09.009

Meeker, H. (2016, September 20). AGPL: Out of the shadows. *Synopsys*. https://www.synopsys.com/blogs/software-security/agpl-affero-gpl-3/

Milinkovich, M. (2020, April 23). *Eclipse Public License Version 2.0 Approved By OSI and Eclipse Foundation Board of Directors | The Eclipse Foundation.* Eclipse Foundation. https://www.eclipse.org/org/press-release/20170829eplv2.php

Montague, B. (2007). Comparing the BSD and GPL Licenses. *Open Source Business Resource*(October 2007).

Morrison, G. (2015). *The Fight For Freedom | Linux Voice*. https://www.linuxvoice.com/the-fight-for-freedom/

Morrison, R. (2018). Energy system modeling: Public transparency, scientific reproducibility, and open development. *Energy Strategy Reviews*, *20*, 49–63. https://doi.org/10.1016/j.esr.2017.12.010

MySQL: Commercial License for OEMs, ISVs and VARs. (2020, March 31). https://www.mysql.com/about/legal/licensing/oem/

No License. (2020, March 31). https://choosealicense.com/no-permission/

Odence, P. (2017, August 15). The quietly accelerating adoption of the AGPL. *Synopsys*. https://www.synopsys.com/blogs/software-security/using-agpl-adoption/

O'Grady, S. (2012). *On the Decline of the GPL: On the Decline of the GPL*. https://redmonk.com/sogrady/2012/02/15/decline-of-the-gpl/#ixzz37g3OOT4K

Open Source Initiative. (2020a, March 5). *The MIT License | Open Source Initiative*. https://opensource.org/licenses/MIT

Open Source Initiative. (2020b, March 8). *Licenses by Name | Open Source Initiative*.

https://opensource.org/licenses/alphabetical

Open Source Initiative. (2020c, March 10). *The Open Source Definition (Annotated) | Open Source Initiative*. https://opensource.org/docs/definition.php

Open Source Initiative. (2020d, April 12). *Microsoft Public License (MS-PL) | Open Source Initiative*. https://opensource.org/licenses/MS-PL

Open Source Initiative. (2020e, May 5). *GNU General Public License version 3 | Open Source Initiative*. https://opensource.org/licenses/GPL-3.0

Perl Foundation. (2020a, April 9). *Artistic License 2.0*. https://www.perlfoundation.org/artistic-license-20.html

Perl Foundation. (2020b, April 9). *Artistic Notes 2.0*. https://www.perlfoundation.org/artistic-notes-20.html

Perl Licensing - dev.perl.org. (2020, May 7). https://dev.perl.org/licenses/

Phipps, S. (2013). *Open source needs to clean up its language*. https://www.infoworld.com/article/2612747/open-source-needs-to-clean-up-its-language.html?page=2

Riehle, D., & Harutyunyan, N. (2019). Open-Source License Compliance in Software Supply Chains. In B. Fitzgerald, A. Mockus, & M. Zhou (Eds.), *Towards Engineering Free/Libre Open Source Software (FLOSS) Ecosystems for Impact and Sustainability: Communications of NII Shonan Meetings* (1st ed., pp. 83–95). Springer Singapore; Imprint: Springer. https://doi.org/10.1007/978-981-13-7099-1_5

Salter, J. (2020). *Open source licenses: What, which, and why*. https://arstechnica.com/gadgets/2020/02/how-to-choose-an-open-source-license/

Schaaf, A [A.]. (2012). *Vergleich von Open-Source-Lizenzen*. GRIN Verlag. https://books.google.de/books?id=Pwcw0W3f4SAC

Schaaf, A [Alexander], Lüchinger, F., & Ellmer, B. (2013). *Open Source-Software: Chance und Risiko für Unternehmen* (1., neue Ausg). ScienceFactory.

Schrape, J.-F. (2019). Open-source projects as incubators of innovation: From niche phenomenon to integral part of the industry. *Convergence: The International Journal of Research into New Media Technologies*, *25*(3), 409–427. https://doi.org/10.1177/1354856517735795

Sen, R., Subramaniam, C., & Nelson, M. L. (2008). Determinants of the Choice of Open Source Software License. *Journal of Management Information Systems*, *25*(3), 207–240. https://doi.org/10.2753/MIS0742-1222250306

SPDX License List | Software Package Data Exchange (SPDX). (2020, February 9). https://spdx.org/licenses/

Stallman, R. (2014). *Why Upgrade to GPLv3 - GNU Project - Free Software Foundation*. https://www.gnu.org/licenses/rms-why-gplv3.html

Stallman, R. (2016). *FLOSS and FOSS - GNU Project - Free Software Foundation*. https://www.gnu.org/philosophy/floss-and-foss.en.html

Stallman, R. (2018). *Don't Say "Licensed under GNU GPL 2"! - GNU Project - Free Software Foundation*. https://www.gnu.org/licenses/identify-licenses-clearly.html

Stallman, R. (2020). *Why Open Source Misses the Point of Free Software - GNU Project - Free Software Foundation*. https://www.gnu.org/philosophy/open-source-misses-the-point.html.en

Stol, K.-J., & Fitzgerald, B [Brian] (2018). The ABC of Software Engineering Research. *ACM Transactions on Software Engineering and Methodology*, *27*(3), 1–51. https://doi.org/10.1145/3241743

Synopsys Editorial Team (2019, October 21). Top open source licenses and legal risk for developers. *Synopsys*. https://www.synopsys.com/blogs/software-security/top-open-source-licenses/

Szulik, K. (2018, April 12). Open source is everywhere: survey results part 1. *Tidelift*. https://blog.tidelift.com/open-source-is-everywhere-survey-results-part-1

Tidelift. (2018). *Tidelift Professional Open Source Survey Results*.

Todorović, A. (2020, April 23). *Open source licensing at GitHub*. https://opensource.com/life/15/7/interview-ben-balter-github

Top Open Source Licenses | Black Duck Software: Top Open Source Licenses | Black Duck. (2020a, March 30). https://web.archive.org/web/20160719043907/https:/www.blackducksoftware.com/top-open-source-licenses

Top Open Source Licenses | Black Duck Software: Top Open Source Licenses | Black Duck Software. (2020b, March 30).

https://web.archive.org/web/20171019180121/https:/www.blackducksoftware.com/top-open-source-licenses

Top Open Source Licenses | Black Duck Software: Top Open Source Licenses | Black Duck Software. (2020c, March 30). https://web.archive.org/web/20180604070734/https:/www.blackducksoftware.com/top-open-source-licenses

Top Open Source Licenses | Black Duck Software: Top Open Source Licenses | Black Duck Software. (2020d, March 30). https://web.archive.org/web/20190115063327/https:/www.blackducksoftware.com/top-open-source-licenses

Välimäki, M. (2005). *The rise of open source licensing: A challenge to the use of intellectual property in the software industry*. Turre Publishing.

Wheeler, D. A. (2017, January 26). *The Free-Libre / Open Source Software (FLOSS) License Slide*. https://dwheeler.com/essays/floss-license-slide.html

Wilson, S. (2015). *Open Source Software Licensing Trends | OSS Watch team blog*. https://osswatch.jiscinvolve.org/wp/2015/02/05/open-source-software-licensing-trends/