Apache 2.0 vs GPL 3.0

Institute for Information Systems and Society

Betreuer: Prof. Dr. Rony G. Flatscher

Verfasser: Hong Gu

19. Juni 2019

Table of Contents

1 Introduction

Free and open source software is an important topic. More and more enterprises use open source software and also try to earn money with it. Even the US Government is deeply involved in FOSS. Some people have tried to make FOSS more popular in order to assists third-world countries. It is reckoned that because of the advantages of FOSS the movement will continue to flourish.

There are two very important FOSS licenses. One is the GNU General Public License. It was drafted by the Free Software Foundation. It‘s very difficult to determine the number of projects that are licensed under GPL however it‘s estimated that the number is very high. The most recent version of GPL is Version 3. It comes with various changes such as improving compatibility with other licenses. However, the two most important changes are in regards to the Digital Rights Management and software patents. (Asay, 2008)

On the other hand, we have the Apache Software License.
It‘s commonly used and a permissive open source software license. It‘s wordier than other permissive licenses such as the MIT License. Apache License it also very important and used by projects such as Apache HTTP server, which is being utilized by over 50% of the web servers in the world. Furthermore, it‘s a very well developed license that does not rely on interpretations from its community in order to foresee potential legal issues. (Sinclair, 2010)

This paper aims to give an overview of these two licenses. Firstly, it does so by explaining basic terms in the FOSS movement. Then both licenses are being assessed in order to figure out a popularity trend and which license may have a brighter future. Lastly, both licenses will be compared. The reader will read about which license is currently dominating and who seems to prefer what license.

2 FOSS

The term FOSS is by now well known by the people working in software development. People who are interested in a programming career have surely heard this term or might have even used open source software. However, developers preferred proprietary software, since the owners of software were able to make a profit from their programs.
Free and open source software are two different terms with different meanings. (Balakrishnan, 2018)

Free and Open Source Software is highly relevant when it comes to the development of Information Systems. One needs to conscientiously consider the conditions of the licenses in order to be able to comply with them.

Free and Open Source software is commonly utilized by individuals and groups. It is seen as a way to develop a software product by the FOSS community and companies that produce proprietary software. Even proprietary software may rely on FOSS code. One methodology to developing software is to include FOSS source code to ones own software and therefore improving its quality and extending its functionality.

However, using FOSS code means that one has to comply with the conditions of the license under which the code has been released. Its highly important to consider complication in compliance scenarios as the licensed FOSS code can have an influence on the whole developed information system.

For example, if one uses the code from the GPL license, this license would obligate the developer to release his whole software under the GPL. Additionally, compliance can be a very complex issue. License have varying conditions and restrictions put on the distribution of licensed code. Some users may not want to comply with the restrictions of a

certain license. Wishing to combine different licenses and changing them throughout a software development project can cause incompatibilities. (Gangadharan et al., 2012)

FOSS enables developers to reuse third-party code in order to produce new software. Legally speaking the new program will be a derivative work. Additionally, their modification and distribution are regulated by copyright law. As a result, FOSS licenses have been drafted to encourage derivative software products. However, occasionally it’s not possible to release software with Foss software components under the same license. This stops developers from reusing these components.

Thanks to FOSS it’s is possible for different programs to interact with each other in order to fulfill a function for users. For example, the Linux based distribution Debian consists of over 18000 different software packages. (Software package is software that is independent of other programs and can be installed in a computer system. They can be applications or distinct software libraries such as libtiff) FOSS software packages reuse quite commonly software packages for other sources.

Consequently, we can view software packages as software components. A FOSS application typically consists of multiple components that have an interactive relationship with each other. So, FOSS software is often created through the reuse of other FOSS code. However, before a component is reused, licensing conditions have to be taken into consideration. Developers who want to make use of FOSS code should learn the licensing terms of the license under which the desired FOSS code has been released. It’s important if the code can be used and how so. Also, it should be kept in mind that some incompatibility issues may arise. (German et al, 2009)

2.1 Free Software

Free refers to freedom in using the software. Free software defines four different freedoms:
The program can be used for any possible purpose. Any restriction such as trial period, or allowing the use for only special endeavors leads to the program becoming non-free.

Secondly, the second freedom allows others to study the source code and modify it to fit their own requirements. If restrictions are put on a program that will hinder understanding the source code or it's modification as well as their use or by adding the requirement of buying additional special license makes the software proprietary and therefore not free.

The third freedom states that copies can be distributed at free will for no cost. If it is legally prohibited to give the software to someone else the program is not free software.

Finally, the fourth freedom says that you can improve the program freely and you can release your version of the free software for the public. This way the community can benefit from it.
(Free Software Foundation Europe, 2019)

Due to these freedoms, the users are able to control the software and whatever function it serves for them. If the program is not controlled by the user the software is non-free or proprietary. Non-free software is in control of the user and the program is controlled by the developer, thus making the program a tool to exercise power over others.

If all four freedoms are provided we can call the software free. The GNU foundation considers all non-free software as unethical. In all situations, the four freedoms also have to be valid for software that has to be used in conjunction. For example, if a program A also starts a program B to

deal with something program B must also be free as program A requires it. However, if A is modified in such a way that it can operate independently from B, program B doesn’t need to be free anymore.

It has to be mentioned, that free software does not mean it has to be non-commercial. Free programs can be commercially used, developed and distributed. Companies being involved in the development of free software is not uncommon these days. These free software are quite important. Regardless of how you have obtained the software either free of charge or by paying for it, you must be able to have the freedom to change, sell and copy this program.

Free software has to grant these freedoms to all users that acquire the software if the user has agreed to the conditions of the license under which the program is released. Discriminating users or a group by saying that they have to pay for the program renders the software non-free. (Free Software Foundation, 2019)

2.2 Open Source

Open source means that people can change and distribute the source code as it is publicly available. The term came from software development. It is used to describe a method to develop computer programs. Nowadays open source covers a lot of values. Open source programs thrive for open exchange, joint participation, transparency and community-driven development.
Open source software is when anyone can take a look at the source code and change it.
(„What is opensource“, n. d.)

Open source does not only imply that there is access to the source code. It also means there are terms in regards to distributing software under open software licenses. Following criteria have to be fulfilled:
There must be free redistribution. It shouldn’t be possible to prevent a party from allocating the software as a part of an aggregate software distribution which consists of multiple programs. There should be no fee for such a transaction.

The programs must come with the source code and has to also allow distribution of that source code. If source code is not distributed with the program, there must be an easy way to retrieve the source code for example through downloading it from the internet without any cost.

Furthermore, the code has to be in a state where it is readable. Code that is tampered on purpose so difficulties arise in reading it is not permitted. Thirdly it must allow modification to it as well as derivations. The new software must be distributed for the same requirements as the original software. (Opensource.org, 2007)

Fourthly a license may forbid the distribution of a modified version of source code only if the distribution of patch files together with the source code is allowed. Derived software may need a different name or version number that distinguishes it from the original software.
Fifthly, it is not allowed to discriminate against people or groups. Furthermore, the license is not allowed to prohibit others from utilizing the program in a certain area. For example, it’s not allowed to stop people from using the program in a business.

Additionally, a license must not be valid for only a product. All parties that receive a distribution of a program licensed under open source should receive the rights listed under the Apache Software License without having to depend on another license.
Moreover, the license is not allowed to restrict other software that is being distributed together with the software under an open-source license. Finally, the license has to be technology neutral.
(Opensource.org, 2007)

3 Software License

3.1 What is a Software License

If there is no contract there will be copyright. Copyright endures for 50 years for computer software after the author has died. Copyright allows us to determine who is allowed to make copies of the software and who may modify and publish these modified versions. Copyright even allows us to prevent everyone from receiving a copy of our software product.

However, if someone wants to license their product that won‘t be their intention. Instead, a license aims to not give property rights over the software that has been distributed to you, but it enables you to use the software under certain limits and conditions. In the case of a contract, more rights and power can be given than the current owner currently holds. The licensor can’t do this with his software product. However, they can restrict and state conditions that have to be fulfilled.

Often companies seek to have more power than the copyright law allows them to have. They do this by licensing their software to others, but under conditions which aren’t stated in the copyright law. (Malcolm, 2003)

3.2 What is not a Software License

A license isn’t the same as a contract. A contract requires that there is an exchange between two parties. This is not the case with open source software. The licensee usually does not provide anything to the person who gives out the licenses. Therefore, no contract is established between a licensor and a licensee. Additionally, a contract must be accepted and this acceptance must be visible in a way. This means you must have had a reasonable chance to consider whether to disagree or agree on the contract and that you have made a clear and visible decision to accept the terms of the contract. (Malcolm, 2003)

4 Apache vs. GPL

4.1 Assessment of GPL

4.1.1 Copyleft

Copyleft licenses require, unlike permissive license, that all derivative work from the original source must also be released under the copyleft license. In essence, copyleft prohibits developers from placing restrictions on users, when they redistribute the software. One could conclude from this that permissive licenses provide freedom to the downstream developers, while copyleft licenses provide freedom to the end users.

Copyleft can be explained with a simple example. If a program is written and redistributed to you, you can edit and use it freely. If you wanted to release version changes and distributed a new software you would have to release them under the same license as the source software. However, you don’t have to release your change under the same license. It only must be compatible with the original. For simplicity sake, the original license is usually kept. GPL is the most commonly used copyleft license. (Cotton, 2016)

4.1.2 GPLv3

The GPLv3 license was published on June 29, 2007. When software is released under the GPLv3 license it automatically becomes free software. Also, all derivative works will become free regardless of who changes the software or what in the code is edited. This means that the software is copylefted. The source code is protected by copyrights, but the rights aren't being used to restrict users in editing, modifying and redistributing the software. Instead, a copyleft license aims to ensure that every user will have freedom in regards to using the software.

GPL3 comes with a few new updates in order to deal with technological and legal advancements. Firstly, there is protection from tivoization. Some businesses have made devices that run GPL licensed software, but they changed the hardware in a manner that allows them to make changes to the software. However, you can't edit the running software. The owner should be the one in control. As soon as a device prevents the user from having control it is called tivoization.

Secondly, a law has been made which forbids free software. For example, the European Union Copyright Directive and the Digital Millennium Copyright Act make it illegal to develop or distribute software that circumvents digital restriction management. Ideally, these laws shouldn't make the rights, that GPL grants, ineffective.
Finally, there is protection from discriminatory patent deals.

Rendering Laws ineffective that forbid Free Software
It's possible to write software that renders DRM ineffective and redistribute it without being prosecuted for bypassing Digital Rights Management measures. (Smith, 2014)

Protecting the Right to modify Software freely
The rights to change one‘s own software is rendered useless if the hardware does not allow it. GPLv3 prevents this from happening by

forcing the distributor of the delivered software to give information on how to install changed software on a device. Therefore, you must be able to get the information that is needed to install GPL3 software.

Protection against Patent Infringement Lawsuits

When a developer distributes software released under GPL3 they also have to give the users patent licenses which allow them to exercise the rights of the GPL3 license. Furthermore, if a licensee attempts to use a patent in order to prevent a user from exercising their rights, the license will be canceled. As a result, downstream developers and users don't need to worry about any future patent lawsuits. GPL3 aims to defend better against patent lawsuits than any other license out there.

With GPL3 there are new compatible licenses such as Apache License 2.0. Other small changes include other ways to convey source code, less source to hand out and an easier way to comply with the terms and conditions of GPL if you have broken them. (Smith, 2014)

License	GPLv1	GPLv2	GPLv3
GPLv1	Yes		
GPLv2		Yes	
GPLv3			Yes
X11/MIT	Yes	Yes	Yes
Old BSD (4 clauses)			
New BSD (3 clauses)	Yes	Yes	Yes
Apache v1			
Apache v1.1			
Apache v2			Yes
Artistic v1.0			
Artistic v2.0	Yes	Yes	Yes
Common Public v1			
Eclipse Public v1.0			
Mozilla Public v1			
Mozilla Public v1.1			

Figure 1: GPL Compatibility. From Compatibility of several FOSS licences with the GPL versions. by Daniel M. German and Jes´us M. Gonz´alez-Barahona, 2009, Copyright 2009 by Daniel M. German and Jes´us M. Gonz´alez-Barahona.

4.1.3 LGPL

LGPL allows developers to combine free software with non-free modules. LGPL is different from strong copyleft licenses such as GPL. It aims to be a compromise between permissive licenses such as Apache and copyleft licenses such as GPL. If a software component distributed under the LGPL license is modified it has to be published under the same license. LGPL is primarily used for software libraries. (Wikipedia contributors, 2019)

Notice has to be given that the library is licensed under LGPL and it's object source code.
Also, a copy of the GPL and the license document have to be provided altogether. (Opensource.org, 2007)

Why the use of LGPL is not advised by the GNU Organisation
Developers of proprietary software can't use GPL licensed code as they would have to release their proprietary code under the GPL. So the advantage is that only free software developers are allowed to use GPL licensed code. However, using GPL in all situations may not be advantageous. There are cases where it makes sense to use a less strict copyleft license. For example, GPL C library was released under the LGPL license. There are plenty of C libraries out there. If that library was licensed under LGPL this would shy away proprietary software developers. There are no benefits for GPL in that, as there are plenty of alternative C libraries. GPL should be used so people will be motivated to contribute to free software projects if they want to use other GPL licensed code in their work. (Free Software Foundation, 2016)

4.1.4 A Closer Look at the Anti-DRM Section

This new section seems rather like a harmless change however, it places strong restrictions upon developers who aim to use GPLv3 code. In fact, out all of the changes that were introduced with Version 3 the Anti-DRM Section is one of the most discussed ones. It essentially says that para-copyright measures shouldn't apply to GPLv3 work. Furthermore, GPLv3 demands that developers are obligated to not forbid circumvention of DRM so the users can fully make use of GPLv3 freedoms. (Asay, 2008)

Additionally, Section 6 says that users must be able to use modified forms of GPLv3 software on devices that are designed to prevent doing exactly that. Moreover, it demands to provide Installation information, encryption keys, as well as any other information required to make use of the modified software, has to be given additionally to the source. (Asay, 2008)

The Free Software Foundation and Richard Stallman the author of this license firmly believe that this section was absolutely necessary in order to protect the freedoms of the users and to deal with the ever-growing threat coming from para-copyright. The term "tivoization" was coined, which basically describes the case with TiVo, where they tried to restrict users freedom. TiVo used GPL software in combination with their well know digital video records. However, they managed to restrict users by adding digital keys to the software and hardware. The users were able to change TiVo's source code, but the edited software would not function properly afterward, because the keys in the software and the hardware wouldn't match. TiVo wasn't the only case. The FSF reported that more and more hardware manufacturers decided to make use of these techniques in order to restrict user freedoms. (Asay, 2008)

Many who share the philosophy of the Open Source Initiative have taken a stance against the Anti-DRM section. Linus Torvald who supports the OSI's way has been criticizing GPLv3 while it was being drafted. His critic

was mainly directed towards the anti-DRM section. In his opinion, this section was written out of following their almost “religious” belief about free software instead of acting rationally. Many see this section as a way to control hardware manufactures through the software license. (Asay, 2008)

Linus believes it is reasonable for a manufacturer to force hardware to use only one version of software licensed under the GPL, because the manufacturer may have only tested only that version. Their concern is particularly justified when it comes to medical devices where it may be necessary to have DRM to forbid not tested versions. Also, where protection of privacy is important DRM has a role to play. Many also say that GPL is controlling where their software is being used.

However, FSF most likely has no intention to control hardware manufacturers but merely aims to protect it’s users freedoms. The FSF concern that developers are using hardware to dodge the restrictions of GPL is legit. Without the DRM section, FOSS could potentially resemble proprietary software more and more. (Asay, 2008)

People supporting the anti-GPLv3 view say that the Anti-DRM section could reduce the contributions coming from corporate businesses. As a result, this could weaken the FOSS movement. Secondly, it is believed that this change would cause less innovation and cooperation. Therefore, FSF wishes to foster more innovation could potentially be crippled by GPLv3 anti digital restriction management if corporate contribution becomes less prevalent through it.

In fact, companies seem to dislike GPLv3 so far because GPlv3 is somewhat “viral”. Viral in the sense that it basically “infects” other software and now with the anti-DRM section it can now even control the hardware that it runs on.
Even though the anti-DRM section may put companies at unease it will most likely not stop enterprises from contributing. The main reason is that

this section won't prevent businesses from using software licensed under GPLv3 internally. Most companies do not distribute their software. Therefore they are not obligated to hand out the source code. This is the case for many companies. (Asay, 2008)

Moreover, there are many companies that make utilize web-based GPLv3 software. They do not distribute the licensed program. For this reason, being there are in no obligation to reveal the source code as well as any installation information. This case also applies to a large number of companies and they help the FOSS development thrive. (Asay, 2008)

4.1.5 Software Patents

Both the OSI and FSF are of the opinion that software patents cripple innovation. Additionally, they think that the protection provided by the copyright law should be sufficient when it comes to software. In GPLv3 a contributor who contributes code licensed under GPLv3 provides a patent license to all licensees.

The contributor only hands out a patent license when he distributes software that contains his modifications. A contributor is defined as a copyright holder who allows the use of a GPLv3 code. Additionally, the patent license is valid for the entire distributed software not only the part which the contributor provided.

The Free Software Foundation believes that patent essentially prevents innovation because innovative developers could face a patent lawsuit from the patent holders. Secondly, patent law gives the owners of patents the right to stop developers from using their intellectual property. This is not desired by the FSF as they think that goes against the vision of free software. Instead, they want open standards and the four freedoms to be exercised. (Asay, 2008)

Then there is the problem that the US government hands out patents quite easily, which makes it even more likely for software ideas to be involved in a patent lawsuit. To make everything worse checking software innovation for patents is harder and even if that has been done many still think that there is a chance that they will violate a software patent. As a result, GPLv3 aims to stop patent holders from diminishing developers freedom. (Asay, 2008)

The possible consequences of the patent section are similar to the anti-DRM section. Many companies may fear using GPLv3. For example, if a company hands out software they effectively give up their ability to file a patent law infringement lawsuit for using the licensed software. Therefore distributing GPLv3 source code could have the effect of making their software patents useless in this scenario.

So, it’s probably better for them to just stay away from GPLv3 in order to protect their patent portfolio. GPLv3 goal was to hinder license proliferation by making GPLv3 popular so that developers would flock to it. However, the section on patents may have the opposite effect. Companies may see GPLv3 as a risk and end up stopping their support and contributions for GPLv3. (Asay, 2008)

Additionally, it should be mentioned that GPLv3 patent section helps to bring some balance in the world of the developers. Large enterprises utilize their patent portfolio to prosecute other or to strike out deals where they come out as the obvious winners. In addition, currently, the system encourages companies to cause unnecessary costs with patent lawsuits. These financial resources could be instead used to foster more innovation and software development. Finally, it should be noted that these patent provisions aim to protect developers from being potentially prosecuted by large corporations. The patent section could potentially facilitate development since developers would be less scared of patent infringement lawsuits. (Asay, 2008)

4.2 Assessment of Apache License

4.2.1 Permissive License

A permissive license is similar to a copyleft license however, also permits proprietary derivative work. There is only a minimum of requirements placed on how open source software component can be used. Furthermore, the licensee has a certain degree of freedom regarding usage, modification, and redistribution of open source code. Open source code can be used in proprietary work and hardly needs any conditions to be fulfilled in order to do so. (Goldstein, 2019)

Many developers prefer permissive licenses because there is more simplicity in reusing source code. Keeping track of source code reused from multiple sources can be of concern. Permissive licenses are compatible with all other licenses. (Hanwell, 2014)

4.2.2 Apache License 2.0

The Apache license 2.0 is an open source software license drafted by the Apache Software Foundation. It’s frequently used and has a big community that supports it. Software licensed under the Apache License 2.0 can be freely used, modified, and distributed however, the terms and condition of the Apache license must be followed.

The Apache license 2.0 is a permissive license. The rights granted by the Apache license are applied to both patents and licenses. One has to bear in mind that the unmodified code has to be published under the Apache license. Also, it is not allowed to name your software product in a way that suggests that the Apache Software Foundation endorses your product.

Derived and modified work can be released under different licenses however, this license requires you to make notices about the modifications you have made in the original software. Apache-licensed software can be used in proprietary software free of charge. (Sass, n. d.)

The Apache License Version 2.0 is not very popular if judged by the number of projects that uses this license. Roughly 2% of the projects on SourceForge make use of this license. However, one shouldn't be misled by these numbers. It's an important license considering that it's being used by Apache HTTP server, which is quite commonly used by web servers. Additionally, projects launched by the Apache Software Foundation and Android utilize the Apache License. Furthermore, the Apache License is well developed and isn't dependent on interpretations of its community to deal with potential legal issues. (Sinclair, 2010)

Copyright License

The Apache License is probably drafted with consideration to the US Copyright Act since the grant language is similar to the rights defined in Section 106 of the US Copyright Act. This license grants rights from every "Contributor". A contributor is defined as someone who has ownership over the work or some parts of it. The idea that the rights granted by a contributor to everyone who wants to make use of their work is not new in open source licensing, but in some cases, this is not being explicitly stated.

A worth mentioning feature of Apache License 2.0 is that it allows sublicensing. The intention behind this might be so it is possible to combine Apache License with other software licenses. For example, if it is desired to put together code licensed under Apache 2.0 with code under a copyleft license, the licensee is obligated to sublicense the Apache 2.0 code under the copyleft code. (Sinclair, 2010)

Patent License

Upon receiving Apache-licensed code the licensee receives both a copyright license and a patent license. This is rather uncommon with permissive open source licenses since patents are rarely mentioned. The patent license only counts for the patents provided by the contributor that is needed by the contribution. (Sinclair, 2010) A contributor is basically anyone who contributes source code to the project. Every contributor grants permission to all licensee to use their patents that are being used in the contribution. This prevents contributors from filing a patent lawsuit against the users of the software. (Kaufman, 2018)

Furthermore, Apache License 2.0 only provides a narrow patent license meaning the patent license will not apply to any changes made in the future. Therefore the grant only is valid for the contribution.

Redistribution

Someone who distributes Apache-licensed software must attach a copy of the Apache Software license and provide a log of the changes made to the edited files. In the section where the redistribution requirements are elaborated the term “derivative work” is quite frequently mentioned. This is probably because the conditions of the distribution of Apache-licensed software only plays a role in regards to derivatives.

Contributions

Any work that makes use of Apache-licensed code without saying so or a license notice counts as code licensed under the Apache License 2.0. However, this is not the case when the developer adds the note “not a contribution”. This clause is important because it diminishes any license uncertainty when it comes to software that is submitted informally. (Sinclair, 2010)

4.3 Comparision of the two Licenses

Generally speaking, developers seem to favor GNU General Public License while companies rather like the Apache Software License more. OpenLogic has conducted some research which shows differing preferences between developers and enterprises. Developers prefer releasing their code under the GPL. However, companies avoid GPL and seem to gravitate towards Apache license or other less restrictive licenses. Enterprises often avoid copyleft licenses out of fear of their requirements and the potential outcome they may have on their intellectual property.

One has to remember that GPL forces anyone who uses GPL code in their project to release their software that makes use of GPL software components under the GNU General Public License. The general opinion on GPL is that it forces you to distribute your program in a way that you have to publish your whole software under the GPL. (Merill, 2011)

Additionally, it is possible that software developers may not actually like GPL. It is speculated that because there is so much GPL code out there people rather choose not to program software components from scratch but rather adapt GPL code to their own needs instead. This forces you, of course, to release your entire program under the GPL.

Apache License is different in that sense. It does not obligate you to distribute your project as GPL does. Companies tend to favor Apache Software License for the reason that the license clearly states what is permitted and what not in conventional legal language. However, one has to bear in mind that there are a few exceptions to a general dislike of GPL by enterprises. (Merill, 2011)

Furthermore, when a developer modifies software that is licensed under Apache and releases it, he doesn't have to publish the modified version of the open source software. Additionally, he is not obligated to send the changes done to the open source software to the upstream developers. The developer just has to attribute the work appropriately and provide a copy of the Apache Software License with the released software. This is obviously a huge benefit for people who use the Apache license. (Ricky, 2011)

4.3.1 Rise of Open Source and Apache-Style licensing

It was common in 2008 to think that the open source movement would suffer greatly without any contribution. However, since then open source growth has been steady. This change in trend is due to permissive licensing and the Apache Software license becoming more popular. Additionally, web giants such as Facebook have been actively contributing to open source projects. This growth is mostly caused by a focus on encouraging developer communities to participate in open source projects instead of trying to make more money. Paradoxically, due to this change, the open source community has made more money and at the same time managed to become more sustainable. (Asay, 2013)

GPL was more important during the early years of the FOSS movement, but with time the trend has shifted towards Apache-style licensing. Developers have been turning their back to GPL. GPL essentially desires that all software should be free. The license should stop any means from turning free source code into a proprietary one. Unlike Apache 2.0, the GNU General Public License places a significant restriction on anyone who agrees to the GPL conditions. If you use GPL code you have to distribute derivative works under the General Public License.

With Apache 2.0 this is not the case. Supporters of Apache think software should be free, but they do not believe that forcing others into releasing free software only is a good decision. Apache licensees are given more choice in regards to what they can do with the licensed software.
As a result, the number of software licensed with Apache Software License has grown substantially. (Asay, 2013)

4.3.2 GPL losing Popularity

GPL's decline could be seen as a rejection of the ideas behind free software and its rigid restrictions. GPL requires all derivative work to be licensed under GPL including software that links to GPL programs. This causes fear among developers, enterprises. It is free however one might have to face the repercussions of how GPL source code may taint one's own software. This is a risk many aren't willing to take.

Around 2005 legal departments tried to figure out how to get involved with open source code without causing any legal issues. GPL was seen as a problem and often Apache was picked for software projects. Since Apache doesn't demand as much from users as GPL does, legal counsel is more willing to embrace Apache. Furthermore, it was easier to finish a project with Apache source code than with GPL-licensed code. Apache simply made it easier to get a program done. (Asay, 2013)

After a while, Facebook started to contribute to open source projects, which was unusual since big companies would usually keep the modifications to open source projects to themselves. They believed that their changes to software would only benefit their competitors and no one else because only the big companies would be able to make use of their programs effectively.

However, web giants have changed their opinion and now believe that how they handle software is what really distinguishes themselves from one and another. Furthermore, Facebook started offering services that were enabled through open source software to others. They didn't have to sell the software anymore, which made it more viable for them to contribute to the open source community.

Other companies have adopted new business strategies namely selling proprietary software or services that are linked together with open source programs. This enables them to contribute to open source projects while making money at the same time. Finally, it can be said that these companies generally favor Apache licenses. (Asay, 2013)

Enterprises tend to favor open source these days and releasing software under permissive licenses such as Apache or MIT. The open source approach is better with software licenses such as Apache. The restrictions placed by GPL on the users and developers is just too much to make it worthwhile to release code under GPL. Even Linus Torvald thinks that GPL's way of scaring freeloaders away is not beneficial. He believes this is how open source simply works. Value is created by contributions to the open source software as well as running the open source software. (Asay, 2013)

4.3.3 Most popular Licenses in 2018

WhiteSource research team has gathered information in order to figure out which software licenses were the most frequently used in 2018. Also, data from 2016 and 2017 was used to put in contrast.
The conclusion of this research is that the trend towards permissive licenses continues to go on, while copyleft licenses and especially GPL are still in a decline.

Apache Software License 2.0 manages to take second place in the category top 10 most popular open source license of 2017. This is no surprise as a permissive license doesn't aim to restrict it's users much. Based on WhiteSource data their finding is that 64% of open source software components are released under a permissive license. Last year this number was at 56%. While permissive licenses are becoming more popular, the usage of copyleft licenses is decreasing. 36% of the most popular 10 software licenses are actually copyleft licenses. (Goldstein, 2018)

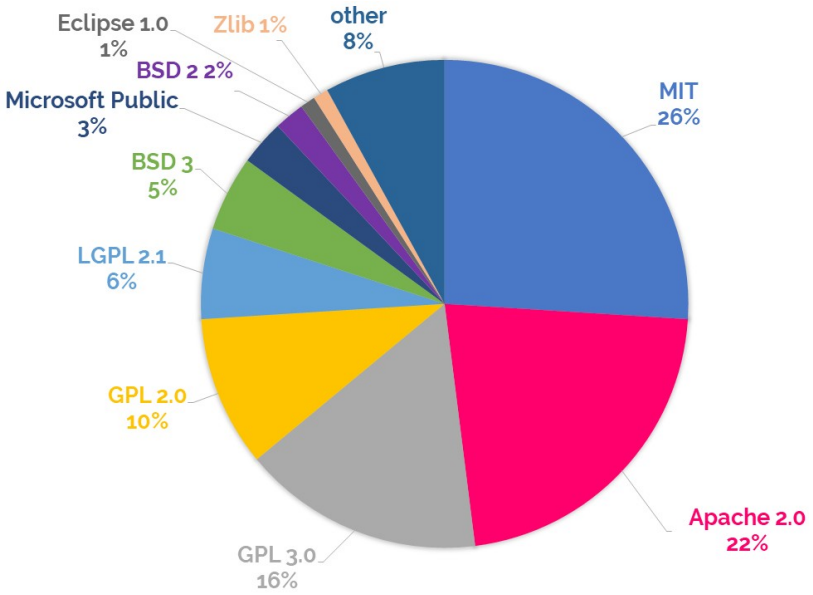

Figure 2: Top 10 Open Source Licenses in 2018. From Top 10 Open Source Licenses in 2018 by Ayala Goldstein, 2018,

https://resources.whitesourcesoftware.com/blog-whitesource/top-open-source-licenses-trends-and-predictions. 2018 by Goldstein/WhiteSource

One explanation for this trend is the growing usage of open source software. In the past few years, the open source community has gained more support from enterprise developers. Companies such as Google and Microsoft are contributing to open source projects. When it comes to license selection for open source software, permissive licenses are favored. (Goldstein, 2018)

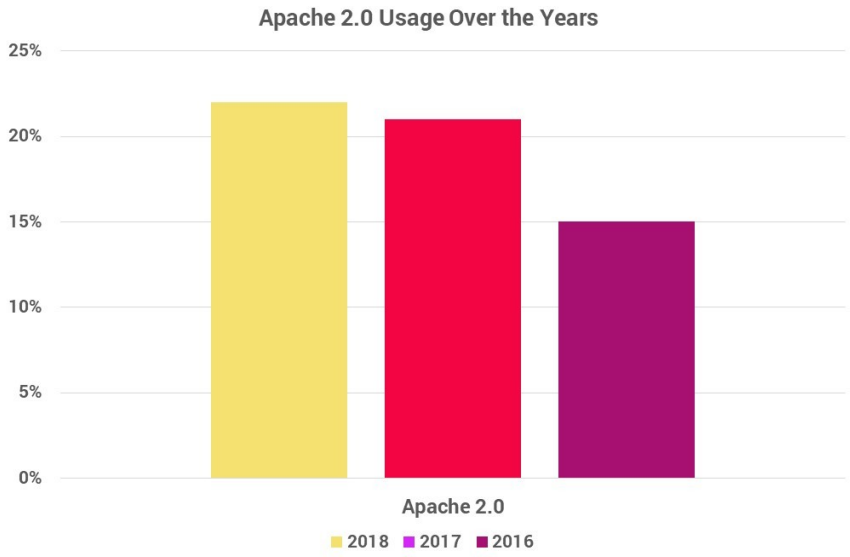

Figure 3: Apache 2.0 Usage Over The Years. From Apache 2.0 Usage Over The Years by Ayala Goldstein, 2018,

https://resources.whitesourcesoftware.com/blog-whitesource/top-open-source-licenses-trends-and-predictions. 2018 by Goldstein/WhiteSource

In 2017 Apache 2.0 managed to make it to the second place in WhiteSource Top 10 List. In 2018 Apache Software License 2.0 managed to grow even further and increased their score of 21% usage to 22%. (Goldstein, 2018)

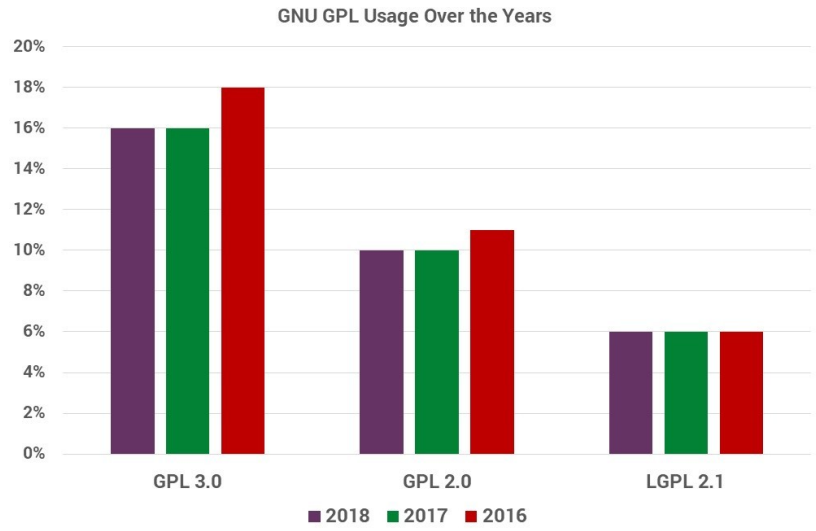

Figure 4: GNU GPL Usage Over The Years. From GNU GPL Usage Over The Years by Ayala Goldstein, 2018, https://resources.whitesourcesoftware.com/blog-whitesource/top-open-source-licenses-trends-and-predictions. 2018 by Goldstein/WhiteSource

GPLv3 and GPLv2 are continuing to decline. GPLv3 is still at place three with 16%, but they had 18% in 2017. GPLv2 is still ranked at place 4 however, it lost one percent in 2018 and fell to 10%.
2017 GPL 3.0, GPL 2.0 and LGPL 2.1 had a combined score of 35% which fell to 32% in 2018. It is reckoned that this decline will continue going on. (Goldstein, 2018)

4.3.4 Choosing a License for Commercialization

Choosing a license for your software project is an important decision to make when you want to begin a project. For most programs, permissive licenses such as Apache or MIT are chosen.
Many are scared of the GPL as it might prevent users from utilizing their program. This concern is not unfounded since Google has banned the use of AGPL3 software. Of course, it is hard to commercialize a product that no one wants to use. (Wang, 2018)

It's not very clear what license to choose if you want to get the most out of your software project. In order to shine some light on this issue data from the index OSS.cash has been looked at.
In November 2018 this index has logged roughly 140$ Billion revenue of open source software companies.

This graph shows how much revenue can be attributed to some licenses.

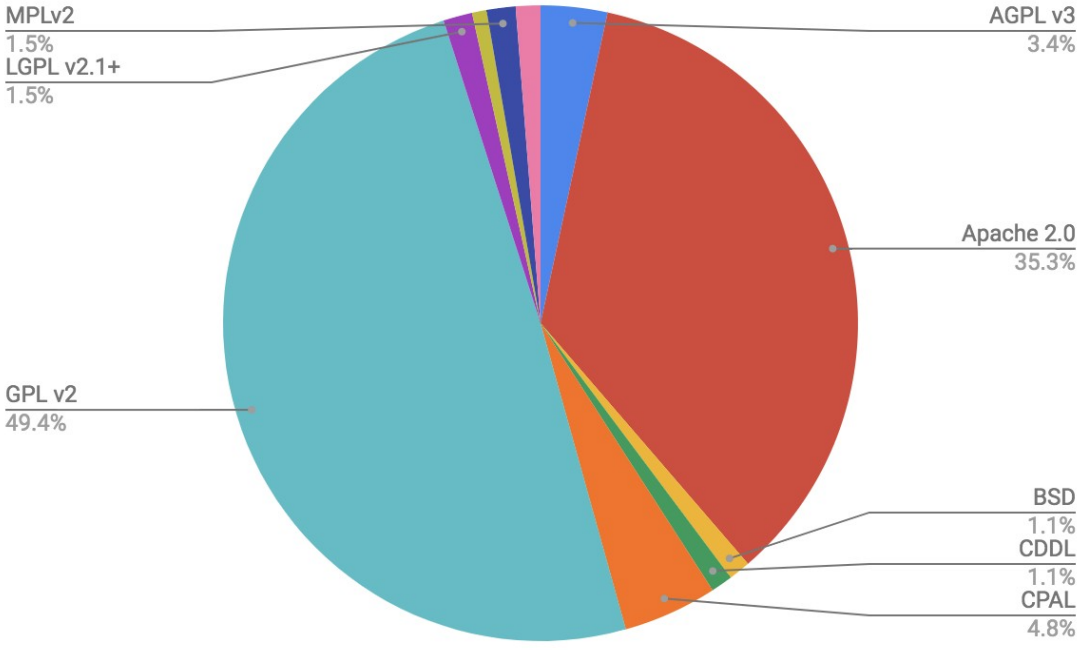

Figure 5: License Valuation. From No Name, by Kevin Wang, 2018, https://fossa.com/blog/which-open-source-license-is-the-best-for-commercialization/. 2018 by Kevin Wang.

Most Open Source Software enterprises make use of permissive licenses, however, companies which develop software which is licensed under a copyleft license seem to generate the most revenue. (Wang, 2018)

If we look at the average value of each license type we will find a different result.

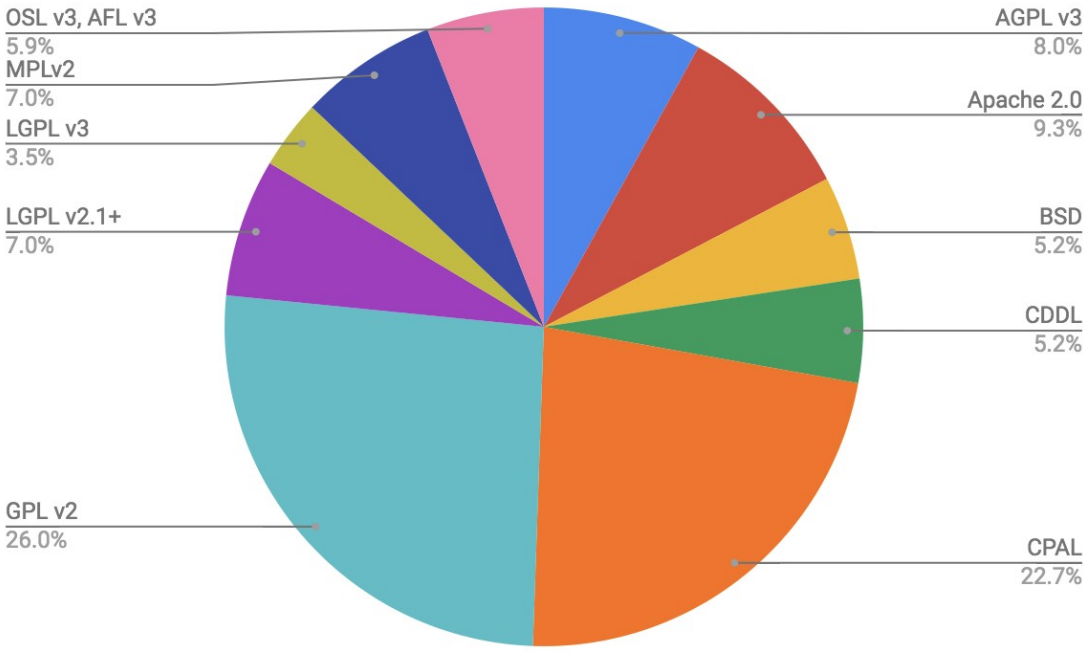

Figure 6: Average License Valuation. From No Name, by Kevin Wang, 2018, https://fossa.com/blog/which-open-source-license-is-the-best-for-commercialization/. 2018 by Kevin Wang.

It's difficult to recommend a default license for every newly started project. The data shown isn't enough as the index only tracks 38 companies. Furthermore, GPL was very common with older generation companies, which have had more opportunities to generate value than others. Roughly 40% of the companies which were founded before 2007 licensed their software under a copyleft license, while only roughly 25% of the companies founded after 2007 used a copyleft license. Additionally, there are outliers such as Redhat which make GPL seem more prevalent.

With this in mind, it should be safe to say that copyleft licenses are viable in a business environment. It’s not possible to say, that restrictive licensing will help your business flourish better however many companies have broken the 100$ million revenue mark with copyleft licensed software. (Wang, 2018)

5 Conclusion

The Apache License is much wordier and therefore allows less room for interpretation. It also added a new definition in order to deal with patents. This software license is written in consideration with the OS Copyright and Patent Law, which allows to clear up things in regards to licensing. (Sinclair, 2010)

Permissive Licenses such as Apache License 2.0 are gaining popularity. They put strict restrictions on its users as copyleft licenses do. It allows you to use, edit and convey open source without asking much in return. More and more people are using open source software components in their code while copyleft is becoming less popular. Additionally, giants such as Facebook and Google are contributing to open source projects. GPL doesn‘t have a good prognosis. It will probably continue to decline. Companies seem to avoid GPL. (Goldstein, 2018)

GPLv3 saw some significant changes. The patent provision and the Anti-DRM section has caused a lot of controversies. However, GPLv3 seems to have found a balanced way to deal with the DRM. It may harm companies who want to implement DRM measures however, users freedom will be preserved by this.

Patent prosecutions are a real threat to developers. The patent section protects them for now especially from companies with sufficient financial resources. The patent term may prevent some companies from using GPLv3 however, it is unlikely that this will be GPLv3 end. However, it will maybe slow down the movement. GPLv3 may not be the most optimal solution however, it is taking steps in the right direction. (Asay, 2008)

6 References

Asay, M. (2013). Q&A. Is Open Source Sustainable?. *Technology Innovation Management Review*, *3*(1).

Asay, C. D. (2007). The general public license version 3.0: Making or breaking the foss movement. *Mich. Telecomm. & Tech. L. Rev.*, *14*, 265.

Gangadharan, G. R., D'andrea, V., De Paoli, S., & Weiss, M. (2012). Managing license compliance in free and open source software development. *Information Systems Frontiers*, *14*(2), 143-154.

German, D. M., & González-Barahona, J. M. (2009, June). An empirical study of the reuse of software licensed under the GNU General Public License. In *IFIP International Conference on Open Source Systems* (pp. 185-198). Springer, Berlin, Heidelberg.

Malcolm, J. (2003). Problems in Open Source Licensing. In *Australian Linux conference*.

Sinclair, A. (2010). License Profile: Apache License, Version 2.0. *IFOSS L. Rev.*, *2*, 107.

Free Software Foundation. (2019, March 20). What is free software? Retrieved from https://www.gnu.org/philosophy/free-sw.en.html

Balakrishnan, Akshay. (2018, July 19). The history of Free and Open Source Software, for the 'Third Generation'. Retrieved from https://medium.com/fossmec/the-history-of-free-and-open-source-software-for-the-third-generation-9997b5f6e73c

Free Software Foundation Europe. (2019, June 12). What is Free Software? Retrieved from https://fsfe.org/about/basics/freesoftware.en.html

What is open source? (n. d.) Retrieved from https://opensource.com/resources/what-open-source

Opensource.org. (2007, March 22). The Open Source Definition. Retrieved from https://opensource.org/docs/osd

Cotton, Ben. (2016, August 12). What is copyleft? Retrieved from https://opensource.com/resources/what-is-copyleft

Smith, Brett. (2014, November 8). A Quick Guide to GPLv3 Retrieved from https://www.gnu.org/licenses/quick-guide-gplv3.en.html

Wikipedia contributors. (2019, June 15). GNU Lesser General Public License. In Wikipedia, The Free Encyclopedia. Retrieved 16:22, June 19, 2019, from https://en.wikipedia.org/w/index.php?title=GNU_Lesser_General_Public_License&oldid=901931174

Opensource.org. (2007, June 29). GNU Lesser General Public License version 3. Retrieved from https://opensource.org/licenses/LGPL-3.0

Free Software Foundation. (2016, November 13). Why you shouldn't use the Lesser GPL for your next library. Retrieved from https://www.gnu.org/licenses/why-not-lgpl.en.html

Goldstein, Ayala. (2019, January 24). Open Source Licenses Explained. Retrieved from https://resources.whitesourcesoftware.com/blog-whitesource/open-source-licenses-explained

Hanwell, Marcus. (2014, January 28). Should I use a permissive license? Copyleft? Or something in the middle? Retrieved from https://opensource.com/business/14/1/what-license-should-i-use-open-source-project

Sass, Rami. (n. d.) Top 10 Apache License Questions Answered Retrieved from https://resources.whitesourcesoftware.com/blog-whitesource/top-10-apache-license-questions-answered

Kaufman, Jeffrey. (2018, February 16). How to make sense of the Apache 2 patent license. Retrieved from https://opensource.com/article/18/2/how-make-sense-apache-2-patent-license

Merill, Scott. (2011, n.d). Developers Prefer GPL, Enterprises Prefer Apache. Retrieved from https://techcrunch.com/2011/05/17/developers-prefer-gpl-enterprises-prefer-apache/

Ricky. (2011, July 18). Why Corporations Favor The Apache License Over The GPL/LGPL
Retrieved from https://digitizor.com/apache-license-vs-the-gpl/

Goldstein, Ayala. (2018, December 13). Top 10 Open Source Licenses in 2018: Trends and Predictions. Retrieved from https://resources.whitesourcesoftware.com/blog-whitesource/top-open-source-licenses-trends-and-predictions

Wang, Kevin. (2018, November 14). Which open source license is best for commercialization? Retrieved from https://fossa.com/blog/which-open-source-license-is-the-best-for-commercialization/

7 Figures