Getting Licensing Right at ownCloud

Case-2016-01-Getting-Licensing-Right

On the morning of October 6, 2014, Holger Dyroff, a co-founder of ownCloud Inc., was on his way to an important meeting. While waiting at a red traffic light, he was outlining for himself the strategy he would be presenting to the other two co-founders of ownCloud. The meeting was with Mr. Frank Karlitschek, who had initiated the ownCloud project and was in charge of development, and Mr. Markus Rex, ownCloud's chief executive officer (CEO).

ownCloud Inc. is a software vendor providing a file sync and share software to enterprise customers and the open source world. The goal of the meeting was to come to a decision about the license of the iOS client, an app that allows users to access their files from mobile Apple devices. Dyroff's responsibilities at ownCloud included the management and governance of intellectual property and licensing. All three co-founders knew that managing intellectual property correctly was critical for the success of ownCloud.

Up until this point, the iOS mobile client had been provided exclusively under a proprietary license, even though the ownCloud software itself was based on open source software. In order to be able to distribute the iOS client via Apple's App Store it was necessary to make the app available under a proprietary license, because Apple did not accept Free/Libre and Open Source Software-licensed programs.

Unfortunately, the iOS mobile client not being open source software meant two things. Firstly, the ownCloud executives learned that the iOS client improved significantly slower than ownCloud's Android client. The Android client was open source software and the community was therefore able to work with it. Secondly, critical comments being made inside the community about the fact that the crowd was deprived of the code of the iOS client were becoming louder and were gaining resonance.

Waiting at the traffic light, Dyroff was weighing in his mind different courses of action and what the associated risks and opportunities would be. Should he advocate open sourcing the iOS client? This would address the problems mentioned above, but wouldn't ownCloud Inc. be losing revenue then?

1. Introduction to ownCloud

1.1 Project and company

ownCloud, Inc., the company, built its products on ownCloud, a Free/Libre, and Open Source Software (FLOSS) project, right from the beginning.[1] When the company was founded in December 2011 it was already able to draw on the work of the ownCloud FLOSS community. By the time the company was started, version 2.0 of ownCloud had been launched.[2] Up until this point, development of the software project had been driven solely by the associated open source community under the management of Frank Karlitschek. It was in January 2010 that Karlitschek had kicked off the ownCloud project at the K Desktop Environment (KDE) conference. Karlitschek had been a FLOSS enthusiast for over 10 years by this time, and had also played several management roles in the KDE community.

Like most successful FLOSS projects, ownCloud came about due to a need for which no other solution so far existed. Mr. Karlitschek was searching for a way to sync(hronize) photos and other files to all of his devices and to share them with his friends and family. Also, he wanted to ensure that large files like videos — which often were too big to be sent via e-mail due to attachment size restrictions — could be sharable without the need to fall back on a physical drive.

At that time, various file synchronization and sharing (FSS) solutions already existed. One prominent example was Dropbox, a company launched in 2007 that had quickly evolved to become an FSS solutions leader, setting a benchmark for all other services that provided similar functionality. Dropbox made it easy for users to store and share data. It also allowed users to sync data between all their devices, enabling them to keep their content and folders synchronized everywhere. Also, Dropbox came with several other practical features such as automatic syncing of pictures that users took with their mobile phones.

For Karlitschek, though, use of such services was out of the question. He wished to conveniently transfer his data, but not at the cost of entrusting his data to a third party service that did not guarantee him certain freedoms (Karlitschek, 2014).

The mainstream FSS solutions, including Dropbox, were mostly offered as Software as a Service (SaaS) and via the public cloud distribution model. In this cloud computing setup, the

1 The term Free/Libre and Open Source Software (FLOSS) was coined with the intention of providing equivalent weighting to both the concepts of free software and open source software. Within the FLOSS community, there is an on-going debate as to which of the two latter should be used to describe software in which the source code is made available and users are assigned the rights to change the software and to redistribute it (Williams, 2002). The term ‘Libre’ was added to ensure the acronym FLOSS stressed the free nature of the software. ‘Libre’ also underlines the fact that the free in ‘free software’ is to be understood with regard to the freedom with which it can be used rather than because it comes gratis, rather as in the case of free beer (Stallman, n.d.). FLOSS, open source software and free software are terms used to refer to the same concept in the majority of cases. Here the terms are all used interchangeably.

2 Please note that the term ‘ownCloud’ is used in this text to designate several things. Firstly, it is used to refer to the software *ownCloud*, which was available in version 8.2 at the beginning of 2016. Secondly, ownCloud is the name of the community project associated with the software. Thirdly, there are also an ownCloud Inc. and an ownCloud GmbH. The GmbH is based in Nuremberg and is a subsidiary of ownCloud Inc. that is headquartered in Lexington, Massachusetts.

software is hosted in data centers controlled by cloud providers, who offer it to multiple customers at the same time (NIST, 2011). Users can access software and storage on demand, for example via a web browser, just like opening a web page. They do not need to install software on their own machine or to provide the necessary computing power by themselves. For example, Google Docs users no longer have to install an Office suite on their computers and when they use Google Docs all data is saved on servers maintained by Google. Customers are therefore not aware of where exactly their data is stored or processed. For them, the data is simply somewhere out there in the cloud. As services are provided via a public network, public cloud architectures are often associated with security concerns.

These security and non-transparency concerns are an important issue for companies that want to provide an enterprise file synchronization and sharing solution (EFSS) for their employees. This can become especially critical in combination with concerns about data privacy, which is a particularly sensitive issue in Europe where companies need to take special care that processing of personal data meets the provisions of the European Data Protection Directive. Due to regulations such as these, it can be crucial for companies to be able to identify where or at least under which legislative framework their data is being processed and stored.

Karlitschek's aim was to provide an intuitive and consumer-friendly FLOSS alternative to Dropbox and other mainstream file sync and share providers — an alternative especially interesting for all those who want to maximize control over their data and have full knowledge of where, in which jurisdiction, and for how long their data is stored.

Thanks to open source, the ownCloud project quickly became a success story. In 2011, just one year after its launch, ownCloud estimated that it had around 350,000 users worldwide (Endsley, 2011). By September 2015, ownCloud Server had already reached the 2.5 million user mark (ownCloud, 2015).

ownCloud Inc. was founded to supplement the ownCloud project in December 2011 by Frank Karlitschek, Holger Dyroff, and Markus Rex. The three German co-founders incorporated in the United States, because of its superior ecosystem for software startups at the time.

ownCloud Inc. is headquartered in Lexington, Massachusetts, close to Boston. The founders found it easier to raise funds from venture capitalists in Boston than in Nuremberg.

Dyroff had extensive experience with FLOSS with regard to business development and became vice president strategic partners at ownCloud Inc. He was also appointed executive director of ownCloud GmbH, the German subsidiary of ownCloud Inc. Markus Rex, with his long background in technology management at FLOSS organizations became chief executive officer (CEO) of ownCloud Inc. Frank Karlitschek remained community leader of the ownCloud community project. In 2012, he additionally stepped into the role of chief technology officer (CTO) to also drive forward product development for the enterprise.

ownCloud Inc. was able to continuously extend its customer base. By the end of 2015, it had more than 300 enterprises subscribing as customers. This was equivalent to more than 900,000 commercial users. They hauled from across 47 different countries.

These impressive figures also resulted in financial success. In January 2016, CEO Markus Rex was able to jubilantly announce that ownCloud Inc. had managed to grow over 100 per cent during the course of 2015, thus giving ownCloud its most successful financial year ever. Also, he gave ownCloud good chances of doubling its revenue again in 2016. ownCloud was thus aiming to realize annual revenue of US$16 million in 2016 (Rex, 2016).

ownCloud had successfully entered a rapidly growing market. The overall worldwide FSS market was forecast, by research and intelligence firm International Data Corporation (IDC), to grow from US$805 million at the beginning of 2014 to a value of US$2.3 billion in 2018 (IDC, 2014).

1.2 Business model

ownCloud Inc. is a single vendor commercial open source software company providing an EFSS software in return for payment through a subscription model.[3] Additionally, it offers services and support to ownCloud users.[4] ownCloud Inc. does not itself host its EFSS solution, so it does not act as a cloud provider.

Exhibit 1 provides a comprehensive overview of ownCloud's business model based on the Business Model Canvas Template of Alexander Osterwalder.

The ownCloud software suite is offered in two different editions. The ownCloud Server Edition comprises only the gratis FLOSS offering whereas the ownCloud Enterprise Edition, which is aimed at large enterprises, comprises extra features. The Enterprise Edition is specifically designed to attract as customers larger organizations in need of a secure EFSS solution that does not come at the cost of their losing control over their data. In order to meet customer demands, ownCloud Inc. has supplemented the software of the community project by adding features that are mainly of interest to larger enterprises.

ownCloud Server is a software suite of client server software that enables users to establish their own file synchronization and sharing service by hosting it on their own hardware. It can be the ideal solution for individuals, a group of users, or for small to medium-sized businesses. The ownCloud Server is developed and maintained via the ownCloud community under the leadership of Frank Karlitschek.

The latest release of the FLOSS server software package can be downloaded for free from the ownCloud project website. It can be used to set up an on-premise cloud solution. A network-attached computer, server, or network-attached storage system (NAS) are all that is required for users to install and run their own FSS cloud concept. Organizations can run an ownCloud server in their own network.

A crucial part of ownCloud's own aspiration and value proposition is providing for a high degree of privacy and control regarding user data, whilst establishing universal file access and the breaking down of data silos. ownCloud manages this by providing FLOSS software for download, so that users can install it in their own infrastructure. *On-premise software* is a term used to describe software which is hosted on the users' own hardware and where it is thus in their sphere of control. Thanks to its on-premise setup, users of ownCloud know where their data is stored. In addition, they have more control over what is happening with their data – certainly more control than if they were to use a FSS solution offered by an external cloud provider in the form of SaaS. Furthermore, it is possible to use the ownCloud external storage option. Thanks to this functionality other cloud services can also be integrated, such as Dropbox. ownCloud's federated cloud feature enables the combined usage and management of

3 The company's strategic setup matches the single vendor commercial open source business model as outlined in a paper by Riehle (2012).

4 Service and support activities tend to be typical of the standard business model employed by open source firms (Fitzgerald, 2006). Additionally, when a company does not fully own the rights to a FLOSS project, it is still possible to create a revenue stream around community open source by offering services and support.

multiple internal and external cloud computing services. Because of this, data stored in different places can be universally accessed via ownCloud's FSS solution without transferring data control to an external service.

The software suite also comprises synchronization clients that allow ownCloud to be accessed from different devices. The desktop client seamlessly integrates with a user's personal computer (PC) and is easy to handle. Mobile clients available for various mobile operating systems are downloadable from the different vendors' app stores. These facilitate file access via mobile phones or tablets and allow data to be continuously synchronized on all user devices. By virtue of its cleanly designed look, ownCloud creates a Dropbox-like user experience.

Due to its modular architecture, the ownCloud Server is extensible via apps that can be downloaded from the ownCloud App Store for free. Developed by members of the community, all apps add extra features to ownCloud. Hundreds of available apps provide for functionality that, in part, goes far beyond FSS alone. For example the app Documents makes possible collaborative editing and creation of text files.

The Enterprise Edition is basically the same software suite as ownCloud Server. As the Enterprise Edition targets organizations with a minimum of 50 users and a sweet spot at 500 users, it has to satisfy special customer expectations. One of its most important features is that it provides for easy integration with the extended system landscape that can be expected in large organizations. For example, the Enterprise Edition offers Microsoft SharePoint and access to Windows Network drives. Additionally, larger companies need to put special emphasis on data management, and a suitable EFSS solution must also be consistent with specific corporate policies. So a high level of control over company data must be ensured. For example, the ownCloud File Firewall allows for fine-grained access control. Therefore the Server Edition is supplemented by special Enterprise Applications developed at ownCloud Inc.

It is not only this extended set of features that customers can unlock when they take out a subscription. For example, they are also provided with all the clients they need for their employees and can create their own branded versions of them. Customers can replace the ownCloud logo with their own. One of the customers of ownCloud is, for example, the German railway company Deutsche Bahn (DB). DB has subscribed to the Enterprise Edition and has installed ownCloud on their network in order to offer an on-premise hosted EFSS solution for their employees called DBBox. The branding of the app with the DB logo means that DB employees do not have the feeling that they are using an external service but a tool exclusive to DB. The ability to brand the clients is especially important for one group of potential customers of ownCloud; these are Original Equipment Manufacturers (OEMs). *OEMs* are companies that redistribute another company's product under their own brand. Blaucloud, for example, is a German cloud provider that hosts ownCloud on its servers in order to offer FSS solutions as SaaS to their own customers. For OEMs like Blaucloud, it can be important to have their own logos on their software so that their customers are able to recognize their products. Providing this option exclusively with the Enterprise Edition helps to make it more interesting for customers than the gratis Server Edition.

ownCloud Inc. generates income by offering its customers subscriptions. The subscriptions are user-based and involve payment of a subscription fee. Whereas the Enterprise Edition is available by subscription only, ownCloud Server is in general free to use. If however, smaller organizations need more support than they can obtain gratis via the community online forum, they also have the option of selecting the Standard Subscription through which they can obtain services and support in connection with the ownCloud Server.

Exhibit 2 shows a comparison of the two different Editions of ownCloud and highlights important attributes.

As a company based on an open source community project, the business model of ownCloud Inc. had, for one thing, to take account of this circumstance while also exploiting this unique characteristic in order to be successful. Firstly, this had implications for the open source development model and the special relationship with the ownCloud community. Almost two years prior to the foundation of the company, the community had laid the cornerstone by starting to implement the ownCloud project. And development continued to be undertaken by the community after the foundation of ownCloud Inc. Secondly, licensing was a special concern that had to be considered when engineering ownCloud's business model.

1.3 Why open source?

Back in 2011, it was not only due to Karlitschek's FLOSS background that ownCloud became an open source project. For him, it was the obvious choice if he wanted to create a FSS solution that could be trusted.

All FLOSS source code can be reviewed or modified by anyone. The software can be freely redistributed. Everyone who has a copy of a FLOSS program can share it with others, with or without modifications. In order to be considered as FLOSS, the source code of the corresponding computer programs must be made available. These features set FLOSS apart from so-called proprietary software, also known as closed source software. Here source code is not made available, and modification or distribution of copies is mostly prohibited in order to maximize potential monetary exploitation. Sharing of proprietary software is mostly seen as an act of software piracy involving copyright infringement unless explicitly permitted by the rights holders.[5]

Open source is also the term applied to the special development model that underlies the creation of FLOSS. The specific FLOSS attribute — that any interested person can alter the source code when desired – allows for a special form of collaborative and decentralized development. Often users contributing to the development self-organize in the form of a community around the project.

Karlitschek knew from his previous FLOSS experience that an open source approach would be more beneficial for realizing ownCloud than the creation of proprietary software.

The first category of benefits Karlitschek intended to exploit for his project are rooted in the special process of open source software development. As the source code is made available, anyone interested enough and with a certain basic set of skills is able to contribute. Also, open source development is a highly cooperative process. Due to the number of helping hands, the productive discussion with like-minded skilled people and the prospect of including different areas of expertise, development will be achieved more rapidly than if a small group of employees only are used.

The open source approach not only speeds up the development process, it can also result in a higher quality and error-robust product, making the software especially secure. This can be explained with the help of what is known as *Linus's Law*.[6] The greater the number of people

5 There are certain forms of proprietary software that also allow for free redistribution of copies. For example, shareware is proprietary software which can be redistributed, but free usage may only be granted for a trial period and users are not free to inspect or modify the source code (Lerner, Tirole, 2003).

dealing with the source code of a program, the more bugs are likely to be detected and resolved and thus the likelihood for security breaches is reduced (Raymond, 2001).

Secondly, the freedom of redistribution and modification makes it easy to incorporate other FLOSS programs into the architecture, assuming that the associated licenses are compatible. By simply reusing codes from other FLOSS projects, certain features can be implemented without having to reinvent the wheel. This characteristic of FLOSS can save valuable development time and speed up projects significantly. For example, one of many programs ownCloud was able to use as a module was the open source web server framework SabreDAV. By incorporating SabreDAV, ownCloud could readily establish communication with the help of the WebDAV protocol. WebDAV is an open protocol through which ownCloud enables file access and synchronization. With FLOSS it is easier to implement the use of open standards and protocols – something that is important for ownCloud to ensure that it is as compatible with as many different file transfer platforms as possible.

Thirdly, the previously specified benefits of open source development accumulate to produce a major additional advantage. The FLOSS character of ownCloud is key to satisfying the high demands that Karlitschek imposed on the project with regard to privacy and user data control. Basically, being open source adds transparency to the equation, as users have the opportunity to look inside the source code and to understand what the software really is doing. It can thus be excluded that programs have unidentified back doors that give others access, unauthorized by the user, to the uploaded data (Stallman, 2009).

Also, vendor lock-in cannot occur in connection with FLOSS, as any missing export feature could be added to enable the transfer of user data to another service.

But the open source concept not only helped Karlitschek and the community to develop ownCloud. It was and is also a strategic asset for ownCloud Inc. A FLOSS version promotes use of a product much more efficiently and cost-effectively than marketing by the company could ever do (Olsen, 2005). Also, ownCloud's FLOSS background successfully helps to distinguish ownCloud Inc. from the roughly 300 competitors in the marketplace. In 2014, the company was classified by market research institute Gartner as a niche player for not coming close to the market share of mainstream EFSS solutions (Arlotta, 2014). Yet, when compared to other FLOSS solutions, ownCloud was the leading open source FSS alternative to the mainstream services, to which all users who have concerns about their data privacy can switch.

6 Eric S. Raymond, a developer who looked at the reasons why open source development is especially efficient, named this effect 'Linus's Law' in honor of Linus Torvalds, who is the creator and long term principal developer of the Linux kernel.

2. Software Licensing

2.1 Intellectual property

Software businesses build on intellectual property. *Intellectual property* is an intangible asset, such as a man-made invention, for which exclusive ownership can be claimed. Authors, artists and inventors are granted exclusive rights in the form of intellectual property rights (IPR) by the legal system, enabling them to use and benefit from their creations. The public does not have the same set of rights as the creator of a particular piece of intellectual property. For example, persons other than the creator of a drawing are not allowed to sell copies of the drawing. If others wish to use something that is considered the intellectual property of someone else, they will need to request the right to use the item from the IPR holder. IPR holders can grant others certain exclusive rights. They can provide permission in the form of a license. Usually, such permission involves payments by the licensee in form of license fees, also known as royalties.

The three main forms of property covered by IPR are trademarks, patents, and copyrights.

Trademark rights protect identifiers (the trademark) that can be used to distinguish products or businesses, such as names, logos, slogans or even tunes that have been created to be associated with a certain company. *ownCloud* is a trademark registered to ownCloud Inc. Because of this, ownCloud has secured the right to prohibit unwanted usage of its product's name and symbol, and to prevent users from being misled by software unlawfully branded as ownCloud.

Figure 1: The ownCloud trademark as registered with the United States Patent and Trademark Office ("ownCloud Logo", n.d.)

Patent rights are IPRs used to prevent unauthorized use of the fundamental aspects of (usually technical) inventions. Patents need to be granted by official patent offices and are granted for a limited period of time only. They represent the most extensive form of IPR protection as they can even serve to prohibit the use of products developed completely separately yet which employ the same idea as that of an invention already patented. This allows patent holders to use monopolistic pricing policies and to earn money by licensing the right of using the patented component to others.

A company wishing to use patented software first has to obtain the licenses provided by the patent holder. In general, software is also protectable via patents, even though patents are only

indirectly applicable to software in the European Union. In the United States, software patents are granted, if they are filed to protect the technical concept of an invention. There is a growing trend towards obtaining software patents, even though there is an extensive global debate on whether, and to what extent, software patents are beneficial. For example, patents can lead to compatibility issues and also overlap with the copyright protection of software. Software patents also add a further level to the complexity of licensing on the basis of copyrights, as such licensing always has to be reviewed with regard to possible violation of software patents (Evans, Layne-Farrar, 2004).

Copyright protects artistic works, such as paintings, songs and books. Copyright applies by default as soon as a creator brings to life a form of individual creative expression in a tangible medium. Basically, owners of a copyright have the exclusive right to distribute their creation, be it by copying it or by giving away slightly modified versions. This is mainly designed to ensure that copyright holders are able to make money from their creations before someone else markets a copy of it. For an artistic work to be protected by copyright law, an author, for example, does not have to request copyright protection. Protection is granted automatically, but only for a certain period of time. Unlike patents, copyright does not protect a complete work. It only covers specific instances produced by creators. If a songwriter composes a song about the sun, only this piece of music is protected by copyright, not the general concept of the sun itself.

Within the context of IPR, computer software is classified as a piece of literature. Software is automatically protected by copyright law as long as the underlying source code is an original work created by the developer. All source code has an initial copyright holder, the owner. The owner maybe the creator, or it may be the legal entity employing the creator, etc. All those interested in distributing copies of the software based on this source code have to request permission from the copyright holder. It is common for developers to transfer the copyright of their work to the company that employs them or to the organization governing the FLOSS project on which the developer is working. Thus, the organizations also have the right to redistribute or license the developer's work. A commercial license to redistribute software again is usually associated with income in the form of royalties. Anyone distributing unauthorized copies of a software program would risk prosecution due to copyright infringement, and could face severe penalties. Copyright holders can set the rules with which others have to comply if they wish to use the software.

All three forms of IPR outlined above allow creators and businesses to protect their market and to generate revenue by issuing licenses for the use and rights of distribution of their products. But those wishing to set the conditions others have to meet must first obtain ownership of the associated intellectual property.

2.2 Free/libre, and open source software licensing

In the context of FLOSS, all three IPRs are important. The FLOSS concept is based on certain principles, such as the facts that everybody is free to study the source code and to modify and distribute the software. At first glance the FLOSS principle seems to be incompatible with the concept of copyright, as copyright law limits the right of redistribution exclusively to copyright holders.

But licensing can resolve this conflict, as licenses allow rights of the copyright holder to be passed on to anyone who wants to redistribute them. The developer or the company that developed the software still retain ownership of the copyright. Yet anyone wishing to make use

of the afforded freedoms needs to exactly follow the licensing terms of the IPR holder. Thus, the concept underlying FLOSS does not conflict with that of copyright — in fact it is even based upon this.

A software license typically defines the rights and obligations put upon the licensee.

FLOSS licenses, for example, typically require that the copyright notice covering the original work is always passed on when a FLOSS product is redistributed. This feature is important in the case of FLOSS projects as developers are attracted to them because they can rely on being able to build their reputation in this way. The intention of developers contributing to FLOSS projects is that everybody should be able to copy their software in order to make it available to everyone else. Also, all interested parties have the right to modify the software, so that anyone can alter it to meet their own needs. In order to make software freely copyable, modifiable, and distributable, FLOSS developers have to explicitly grant these rights to others. FLOSS licenses are the tool that generally makes this possible. Without these licenses, the rights of making and distributing copies would remain exclusive and the idea of FLOSS would thus not exist.

A special document, the Open Source Definition, details further characteristics that software licenses have to meet so that the associated software can qualify as open source. The document is maintained by the Open Source Initiative (OSI), an organization founded in 1988 by open source advocates. The Open Source Definition explicitly specifies free redistribution as a requirement, stating that FLOSS licenses must not restrict the passing on of software to others. All FLOSS licenses therefore have to ensure that the licensee is allowed to redistribute the software. Also, the free distribution principle means that the licensor does not have the right to demand royalties or other payments in exchange for granting FLOSS licenses (“Open Source Definition”, n.d.). This implies that no business model that would generate revenue in the form of income from licenses granting rights of usage or distribution of single copies would be applicable in the case of FLOSS, mainly because businesses would not be allowed to restrict redistribution and customers would be able to give away the software for free. After a certain period, nobody would be willing to pay a business the fees it would be demanding.

That, again, represents a major difference to field of proprietary software, where a typical business model is to generate income by selling the right of use of a single copy of the software. Customers do not purchase the software per se but the license to use their copy together with the medium the software is supplied on, as for example a CD-ROM. Usage licenses are usually granted in the form of End User License Agreements (EULA), which typically exclude the freedoms granted by FLOSS licenses. Proprietary licenses usually provide only n permission for users to use the single copy of a software they have obtained while excluding any further forms of usage. Proprietary licenses therefore aim to limit usage rights whereas FLOSS licenses aim to entitle users to rights otherwise exclusive to the copyright holder.

There are two subclasses of FLOSS licenses. It is important to distinguish between permissive and reciprocal licenses.

Permissive licenses impose some conditions on the distribution of the software to which they are attached. Their primary purpose is to grant users the typical FLOSS freedoms so that they can freely use the code, make modifications, and redistribute the code and programs created from it. Typically, permissive licenses exclude, via warranty disclaimers, all liabilities for any malfunction or damage caused by the software. Finally, they include an attribution clause that requires that upon use, the copyright holders’ notice be included in the redistribution. These clauses ensure that the sources of the FLOSS components can be identified and that the origi-

nal licensing conditions and disclaimers are always distributed together with any derivatives. Due to the attribution clause, users wishing to redistribute software have to incorporate the original copyright notice and the original licensing text.[7]

The MIT License, originally conceived at the Massachusetts Institute of Technology (MIT), is the most popular FLOSS license ("Top 20 Open Source Licenses", n.d.). Its is also one of the earliest FLOSS licenses, originally designed to facilitate in a concise manner the sharing of software for academic projects (Laurent, 2004).[8]

Similarly, another of the early forms of license also had its origins at a university; this is the Berkley Software Distribution (BSD) license family.[9] The 2-Clause BSD License is very similar to the MIT License. The 3-Clause BSD License has an additional requirement in that next to the required attribution the names of the original copyright holders may not be used for promotional purposes.[10]

The Apache License 2.0 is maintained by the Apache Software Foundation as a licensing tool for Apache projects, including the Apache HTTP Server.[11] It focuses on software development by large communities. The Apache License 2.0 includes a special paragraph that also transfers usage rights for software patents if the open source code distributed under an Apache License 2.0 also incorporates patented ideas. Even though it bears the Apache name, it is also used by many FLOSS projects outside of the Apache world which want to open-source their own software under the same terms.

Reciprocal licenses share the attributes of permissive licenses. They also aim to grant users extensive copyright usage rights and on the other hand guarantee attribution of the source and its copyright holder, as well as the original license. What is more, reciprocal licenses ensure that FLOSS source codes remain open and free in the future. This effect is enabled by an additional attribute of reciprocal FLOSS licenses that has been designated copyleft.[12]

Copyleft adds more restrictive requirements to the rights to redistribute software. Basically, all those wishing to pass on software that is attached to a reciprocal license have to use the same license for redistribution. This also applies if developers modify the code in order to add any features or to create interoperability with another program (Jaeger & Metzger, 2011). The outcome of modifications is referred to as a *derivative work*, comprising a piece of the original software and some kind of extension. Derivative works must be distributed under the same reciprocal FLOSS license attached to the original element used.

---

7 In contrast to reciprocal licenses, in this case the licensing conditions of the original source code are stated for information purposes only and do not restrict users when it comes to choosing a license for redistributing copies of permissively licensed software.

8 The MIT License consists of three short paragraphs only: a copyright notice, a permission notice and a warranty disclaimer. It can be viewed on the OSI website (https://opensource.org/licenses/MIT).

9 The template of the 3-clause BSD license can be found together with a comparison with the 2-clause text at https://opensource.org/licenses/BSD-3-Clause.

10 Even though it is a very popular license, the third clause is known to be associated with incompatibilities with other FLOSS licenses, such as the GPL family (Laurent, 2004).

11 The Apache License 2.0 is already a more extensive legal document and can be found on the website of the Apache Software Foundation (http://www.apache.org/licenses/LICENSE-2.0).

12 Copyleft was conceived and named by Richard Stallman when he drafted the first version of the GPL. The name indicates that no further restrictions may be applied to copies and modifications made from the original than those imposed by the original license. It is, of course, also a wordplay on 'copyright' (FSF, n.d.).

Furthermore, copyleft affects combined works. A typical way to combine code components is via linking. Here, code components in the form of libraries or plug-ins are not incorporated but combined via links to the single sources. When using a reciprocally licensed component to create a combined work, there is a high likelihood that it has to be licensed under the same reciprocal FLOSS license. Copyleft is sometimes considered to be viral, as reciprocal FLOSS licenses seem to spread quite aggressively, because they are applied to other code components. However, collective works make it possible to combine reciprocally licensed components with other licensed components and to issue the result under a freely selectable form of software license. *Collective works* are created when different software elements are combined to form a larger software package, as is the case with the various Linux products. The single programs incorporated in a collective work can communicate with each other but are not linked. With this set up, it is for example possible to distribute a software suite that consists of proprietarily licensed as well as reciprocally licensed software components.

Exhibit 3 provides an overview of the effects of copyleft on software with reference to the various possible modification and combination types.

The most popular reciprocal license family evolved around the General Public License (GPL), which also was the first copyleft license.[13] Richard Stallman had originally created the license in order to use it for the software components of the GNU project.[14] Version 2 of the GPL is the second most popular FLOSS license after the MIT license ("Top 20 Open Source Licenses", n.d.). ownCloud uses Version 2 as well as Version 3 of the GPL. Both versions have in common the fact that no further restrictions than those stated in the GPL may be added to a GPL-licensed code in the process of redistribution. However, GPLv2 was such that it was incompatible with many other licenses. A revised version, GPLv3, was created to provide for greater compatibility with other licenses. For example, GPLv3 is compatible with the Apache license 2.0. Code under the Apache license 2.0 may be combined with code covered by GPLv3. But because of the copyleft element in the GPL, the resulting derivative or combined works must be published using the GPLv3 license. In contrast with Version 2, GPLv3 includes a patent clause that passes on rights with regard to any possible patents of the licensor if licensees need to use these in order to take advantage of the rights granted to them through GPLv3 with regard to copyrights.

The Affero General Public License Version 3 (AGPLv3) is a license based on the concepts of GPLv3. It has one additional condition that goes beyond the other GPL licenses. GPLv2 and GPLv3 assume that software is distributed in the form of copies — copies, for example, on a CD-ROM or available via downloads — that are obtained by the eventual users. The conditions of the regular GPL versions are not triggered until such a case of distribution occurs. For example, no source code has to be made available without redistribution. If, however, software is made accessible as SaaS by cloud providers, no actual copies are distributed. This means, in the case of GPLv2 and Version 3, that the cloud providers are not bound to provide the source code of any potential modifications made by them. The AGPL was designed to close this copyleft loophole by adding a condition for the usage of programs distributed via computer networks. Cloud providers offering an AGPL-protected software on their server are thus also required to provide the equivalent source code. ownCloud strategically provides the server core under AGPLv3 license in order to have an additional incentive that would encour-

13 The different versions of the GPL license family can be downloaded from the GNU website (http://www.gnu.org/licenses/).

14 In the GNU Manifesto, Richard Stallman outlines his concept of the project, his main aim being to develop a free software operating system (http://www.gnu.org/gnu/manifesto.html).

age users to pay for the Enterprise Edition. OEMs wishing to use the ownCloud Server Edition to set up a FSS service hosted on their network resources are, due to the AGPL, required to open-source all modifications they may have made to the ownCloud Core. They can offer a derivative work based on ownCloud without open-sourcing their source code only if they use the Enterprise Edition, where the Core is covered by the ownCloud Commercial License.

The Lesser General Public License (LGPL) was designed to provide protection to code libraries. The LGPL provides a weaker form of copyleft protection that applies to derivative works only and not to combined works. As a result, programs using LGPL-protected libraries can be licensed freely without the copyleft obligations, just like permissive licenses.

2.3 Licensing compatibility

Code components issued under different licenses may not always be legally combinable due to the differing licensing terms. When complying with the terms of one license means violating the terms of the other, the situation can only lead to copyright infringement. The GPLv2, for example, has proved itself to be a license that is incompatible with many other licenses, even other FLOSS licenses (Lindberg, 2008). This can be largely attributed to the clause that no further conditions are allowed to be added to the redistribution of GPL-licensed code. Incompatibility with the GPLv2 can easily become an issue in the case of derivative or combined works. For example, the Apache License 2.0 is incompatible with GPLv2 because the Apache License 2.0 has certain provisions designed to circumvent any threats to free redistribution arising from software patents that are not covered in GPLv2. It is not possible to create derivative or combined works with Apache-licensed code and GPLv2-licensed code. GPLv3, on the other hand, has been supplemented with a paragraph containing the same restrictions concerning software patents, meaning that GPLv3 and Apache License 2.0 are compatible.

Due to copyleft, all derivative and sometimes also combined works resulting from the combination of reciprocally licensed code components must again be issued under a reciprocal license. This becomes especially interesting in the case of a proprietary software project in which FLOSS components are to be incorporated, but where the aim is to retain the right to publish the result under a proprietary software license. In this situation, reciprocal licenses are not compatible with a proprietarily licensed code. The viral character of copyleft can also jeopardize the publication of the end result under a proprietary license, certainly if the result is a derivative work and with a high likelihood if it is a combined work. The safest way would be combine it in a software package in the form of a collective work. This would reduce the risk of the viral effect and ensure the best level of compatibility. The use of LGPL-licensed libraries in a proprietary software project would also exclude the risk of losing the right to freely chose the form of license. The weak copyleft effect of the LGPL does not apply to combined works. Permissive licenses, on the other hand, can be combined with proprietary licenses freely, as even a derivative work can be licensed without any restrictions. Exhibit 3 also shows how combinations with reciprocally licensed FLOSS can impinge on flexibility.

2.4 Dual and multiple licensing

The issue of software licensing can become particularly complex when FLOSS is involved. At ownCloud, licensing is deeply interwoven with the company's business model. This creates complexity but also offers considerable opportunities, such as, for example, by allowing for the use of open source but at the same time charging for licenses.

Dual licensing provides a way of issuing one software product under two different licenses. Dual licensing requires full control of the copyright of the software module that is to be distributed under the two licenses (Olson, 2005).

Dual licensing unlocks new business opportunities for organizations with regard to FLOSS projects because it allows them to distribute a single software product under a FLOSS and a proprietary license. It is thus key to the generation of software licensing revenue by businesses employing the FLOSS concept and to making possible single vendor commercial open source firms (Riehle, 2012).

Software can also be offered under more than two different licenses; this is called *multiple licensing* (Rosen, 2005). Assigning more than one license to a program is not only the basis for the single vendor commercial open source business model. Dual licensing can also be important in the case of software development in order to avoid complications with non-matching licenses of single software modules that are to be combined. Developers who want to reuse parts of code but are unable to do so because of their licenses can approach the copyright owners and ask whether they would be prepared to allow their code to be issued under an additional license that would make usage possible. This is also known as *relicensing*.

3. Complexity of Licensing at ownCloud

3.1 The licenses in use at ownCloud

ownCloud makes strategic use of the various kinds of licenses in order to achieve different objectives. Alongside the already described FLOSS licenses, two types of proprietary licenses are also important for ownCloud.

The ownCloud Commercial License is a proprietary license specifically designed to fit the dual licensing strategy of ownCloud Inc. It is the most important license used in connection with the Enterprise Edition. Dyroff and the other co-founders created the Commercial License in order to be able to offer a proprietary license to their customers that would nevertheless include as many of the FLOSS-typical freedoms as possible. The Commercial License of ownCloud grants customers the right to examine the underlying source code and to modify it. However, it is still a non-FLOSS license as it only allows these freedoms to paying customers and restricts redistribution.[15] Customers are not allowed to pass on copies. These license attributes are particularly beneficial for customers as it enables them to seamlessly integrate the ownCloud EFSS solution into their systems.

In addition, customers do not have to be afraid of accidentally open-sourcing their intellectual property because of effects of copyleft, which could happen if they were to use a FLOSS

15 The ownCloud Commercial License can be viewed on the website of ownCloud Inc. (https://owncloud.com/de/licenses/owncloud-commercial/).

product instead. In addition, ownCloud EULAs grant usage rights for the proprietary version of the Android client and of the iOS client.[16]

Exhibit 4 provides a summarized overview of the most important licenses at ownCloud.

3.2 The code components of ownCloud

ownCloud's product architecture, consisting of various code components, played an important role for the three co-founders when setting up their licensing strategy. Each edition of ownCloud basically has three main component types: the core module, apps, and three different clients.

- The ownCloud Core represents the heart of the software suite and is the server program that the users install on their network in order to host their own FSS service.
- The modular design of ownCloud allows users to supplement the core as required with extra functionality in the form of apps. The apps for the Server Edition are offered through the ownCloud App Store. These apps can be developed by all community members who want to integrate any additional features in the core. ownCloud does not necessarily have any control over the copyright of an app.
- The clients are applications that users can install on their various devices to access the core and to provide synchronization with the core. When using clients, users do not have to log into their own FSS server via their web browser. Furthermore, the clients allow for additional functionality on the device side, such as seamless integration in the filing structure of the operating system.

As part of the dual licensing strategy, all these single subcomponents can have more than one license. The license attached to a component can differ according to whether it is used in the Server or in the Enterprise Edition.

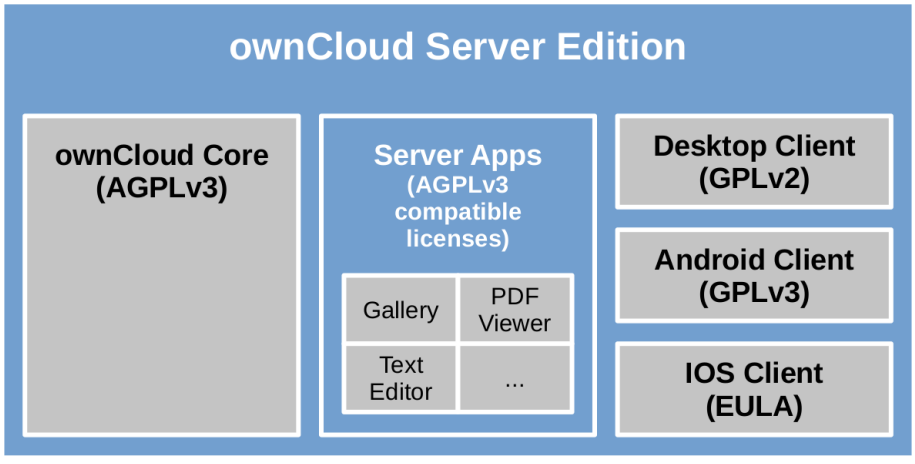

Figure 2: Overview of the components of the Server Edition and the attached licenses

The components of the Server Edition — being ownCloud's FLOSS offering — are largely issued under FLOSS licenses. As AGPLv3 is used for the ownCloud Core of the Server Edition,

16 ownCloud's EULA for the Android and the iOS client can be viewed on the website of ownCloud Inc. (https://owncloud.com/de/licenses/owncloud-android-application/).

the whole edition is considered to be licensed under the AGPL. The only component of the Server Edition not under a FLOSS license is the iOS client.

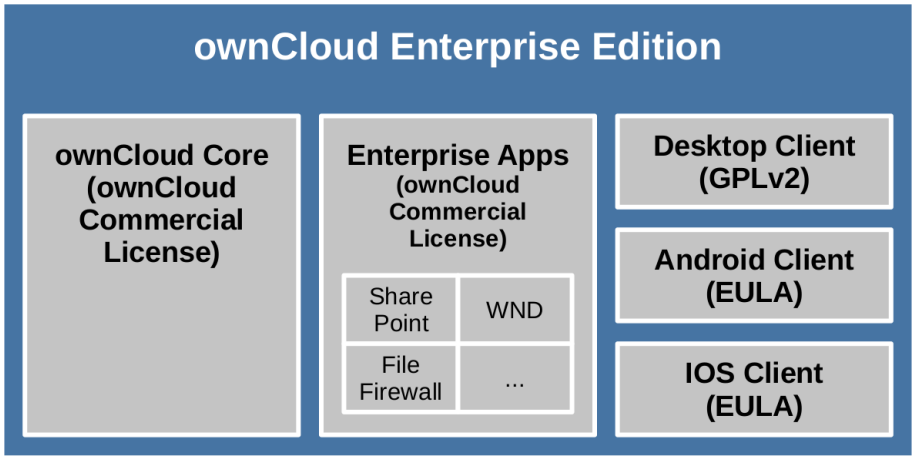

Figure 3: Overview of the components of the Enterprise Edition and the attached licenses

As the Enterprise Edition targets potential EFFS customers who will be required to pay fees in order to be able to access it, the components are for the most part provided under proprietary licenses. However, the technical setup of the single components does not differ from that of the Server Edition, with the exception of the apps offered. In order to position the Enterprise Edition on the EFFS market, the Enterprise Apps have been especially developed to meet the demands of large enterprises. The Enterprise Apps therefore differ from the Server Apps on offer. In addition, the Enterprise Apps are distributed together with the core of the Enterprise Edition and do not represent optional add-ons, as they do in the case of the Server Edition.

3.3 Licensing and the ownCloud community

FLOSS licenses are crucial to FLOSS and open source software development. Without the explicitly granted rights, ownCloud could not have been shared freely and no one would have been able to contribute to the ownCloud community project when it was launched back in 2010. Karlitschek wanted to make sure that ownCloud would not only start as FLOSS, but remain freely accessible to all interested parties in the future. Due to its copyleft attributes, the GPL license family is especially well suited to making sure that this is the case.

Karlitschek chose the AGPLv3 for licensing the core because the ownCloud core is a server software application that users could utilize to offer their own hosted FSS service. Only the AGPL's network reciprocity feature was able to ensure that, within this possible setting, the source code of possible derivatives would remain openly accessible. In addition, use of AGPL and GPL sent out a powerful message that managed to attract like-minded people wishing to contribute to a durable FLOSS alternative to mainstream FSS services. For Karlitschek, it was also important that ownCloud would be compatible with the KDE solutions and the overall Linux family.

3.4 Licensing and ownCloud software development

Just like most other software projects, ownCloud reuses code written by others. This is supported by the common software design technique of modular programming. Software is de-

signed on a modular basis, with the functionalities being segregated in different modules. This makes code parts more easily replaceable and facilitates the process of creating new software by recombining the single components. Due in particular to the free redistribution requirement, many FLOSS components are available for reuse. Yet developers have to respect the terms of the different licenses attached to the single programs. When the different modules are combined, the compatibility of the single licenses also has to be borne in mind.

ownCloud exploits the benefits of FLOSS components. Some programs are offered solely as modules for integration in other programs in order to supplement these with a certain functionality. SabreDAV, for example, is such a module that allows ownCloud to use the open WebDAV protocol for file access.

A particularly efficient way to reuse code is offered via libraries. Using libraries means that pre-written code does not have to be duplicated and physically incorporated into different parts of a program in order to make it work. Instead, the program can simply call the library to retrieve its functionality whenever this is needed. Thus, the code of the library can be provided in addition to the program instead of being actually incorporated. ownCloud uses many different libraries as a basis for its different components.

Whenever ownCloud falls back on code created by others, the licenses attached to the single modules and libraries have to be evaluated. The situation when developers evaluate and accept the licensing terms of desired code modules can be referred to as the *in-licensing of software* (Rosen, 2005). Modules that are used in this way are referred to as *third party software*. For example, when ownCloud uses a LGPL-licensed library developed by another FLOSS project, this can be described as third party FLOSS. When in-licensing, developers need to bear in mind the possible consequences that acceptance of the licensing terms will have on the further development and — most especially — the distribution of their software.

3.5 Licensing and the ownCloud business model

With its single vendor commercial open source business model, ownCloud Inc. generates revenue not only by providing services and support. One important revenue source are the subscription fees paid by the businesses the firm has been able to secure as customers of the Enterprise Edition. Enterprise customers pay subscription fees for permission to use the Enterprise Edition because that is the only way to gain access to additional value in the form of extra functionality, brandable clients, and the avoidance of licensing issues.

In order to realize the planned business model, the co-founders of ownCloud had to establish a dual licensing strategy. On the one hand ownCloud Server was kept under the AGPL license and, as such, offered all the FLOSS-typical freedoms. On the other hand, the company licensed ownCloud Server under their own created proprietary license, allowing ownCloud Inc. to add exclusive extra features and to offer the software in the form of the Enterprise Edition under the ownCloud Commercial License.

This complex dual licensing strategy shows that the co-founders had to put considerable thought into the process of choosing the best licenses for their own software — a process also known as *out-licensing* (Rosen, 2005). Their range of choice, of course, was extensively determined by the degree of freedom provided by the terms of in-licensed software. Most especially, if a reciprocal FLOSS licensed module were modified, almost no freedom of choice would remain. In principle, the licensor first needs to meet the obligations imposed by the licenses of reused third party components before making any own plans. Another major factor

is the degree to which the owner intends to share IPR — ownCloud's decision to establish a proprietary software by using the dual licensing strategy indicates that this factor was a major determinant.

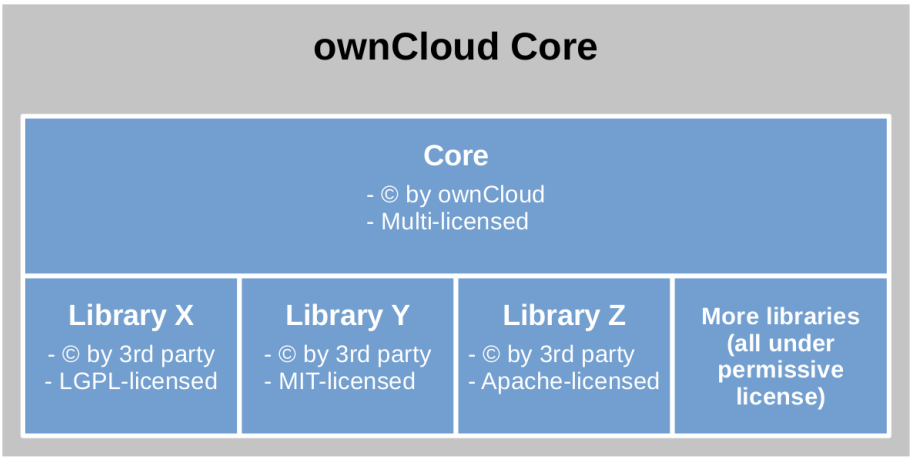

Figure 4: Setup of the ownCloud Core

For its core, ownCloud also in-licenses several components and applies the dual licensing strategy for out-licensing. Figure 4 provides insight into the next deeper layer of the core. On this level it consists of the core program developed by ownCloud that works with different code libraries, all providing certain functionalities needed by the core. The code libraries used represent third party open source code for which ownCloud does not hold the copyright.

The reciprocal AGPL was selected to make the software available to open source users. This was considered important in order to incentivize ownCloud Server users to change to the Enterprise Edition, which requires payment of a subscription fee. Hence customers who plan to make modifications to ownCloud, or to incorporate it into their existing software, may prefer to pay money for the Enterprise Edition. This is due to the fact that, in the case of the gratis Server Edition, the copyleft requirements of the AGPL and GPLv2 might require customers to publish their code under the same reciprocal license. This means they would need to open-source code they prefer to keep closed. When using the Enterprise Edition instead, customers are protected — by ownCloud's Commercial License — from the viral effect of reciprocal FLOSS licenses, but also have the freedom to make modifications or additions.

As the example of the ownCloud Core shows, different priorities apply with regards to in-licensing. ownCloud prefers to use permissive licenses for in-licensing components, as they do not have to worry about these licenses imposing restrictions on their dual licensing strategy. The only reciprocal license ownCloud uses for in-licensing is the LGPL, as linking to LGPL-licensed libraries does not preclude a dual licensing strategy assuming these libraries are not modified by ownCloud.

However, permissive licenses or the LGPL do not meet ownCloud's requirements for out-licensing. In the case of the Server Edition, they provide insufficient protection as the code they are attached to can easily be incorporated into proprietary third party software. On the one hand, permissive licenses cannot guarantee that derivatives of the Server Edition remain open source. On the other hand, they allow competitors to reuse the innovation developed for ownCloud without any conditions being imposed. So when out-licensing, only reciprocal FLOSS licenses can be used strategically as a barrier, the result being that larger enterprises prefer to

avoid any risks by subscribing to the Enterprise Edition. AGPL has been chosen specifically with potential OEM users of the Server Edition in mind. If the core of the Server Edition were to be offered under a GPL version, OEMs could exploit the loophole in the copyleft protection by not directly distributing the software and offering it as SaaS. But the network reciprocity feature of the AGPL guarantees that OEMs are also required to open-source any modifications they might make. Also, the AGPL provides an incentive for this special customer group to switch to the Enterprise Edition instead. Here, the Commercial License allows users to make all necessary changes, but does not require them to open-source the respective code.

The co-founders had to tackle a final complexity in order to be able to out-license the single components of the FSS software suite under the dual licensing strategy. Licensors need to have ownership of the copyright of components whose rights they wish to grant to others, or at least have permission to redistribute the software. Companies need to establish this control over their product in two ways. Firstly, they have to make sure that they have the right to use and distribute all code contributions from all developers involved. This can also have an impact on open source development. Usually, all developers planning to furnish code for the project are required to assign the rights of use of copyrights and patents to their employers or organizations governing the project. ownCloud Inc. provides for this by requiring all developers who want to contribute to the development of the core or clients to sign a contributor agreement.[17] In addition, companies need to make sure that they have permission to redistribute and license possible third party software components reused for the software. This is already has to be borne in mind when in-licensing components.

Because of the enormous impact of licensing on a company's success, software companies need to be in control of in-licensing decisions on the one hand and out-licensing on the other.

They need to have their developers trained so that they are capable of making decisions in accordance with the wishes of their employers. It is also important to establish policy guidelines (Helmreich & Riehle, 2012).

At ownCloud, key licensing decisions are always made by the board, just as the co-founders together developed the basic licensing concept for ownCloud Inc. Regarding in-licensing of third party FLOSS, only modules and libraries with permissive or, at most weak reciprocal licenses are accepted. On the other hand, as part of their dual licensing strategy the co-founders made the decision that, with regard to the FLOSS modules, robust copyleft licenses would always be used for distributing their product. This ensures that the software always stays FLOSS, which was the primary goal of Karlitschek in the early days of the ownCloud community. For the company too, however, this is of strategic relevance as only a GPL or similar can prevent proprietary companies from evolving into competitors using the groundwork put in place by ownCloud.

17 The ownCloud Contributor agreement can be viewed on the website of ownCloud Inc. (https://ownCloud.org/contribute/agreement/).

4. The iOS Client Licensing Issue

4.1 The mobile clients of ownCloud

The various clients are key components of the ownCloud product, as only they facilitate a genuinely mobile file sync and sharing experience that allows users to access their files on one device in the currently updated status as last saved on another. It took ownCloud almost two years to get file synchronization up and running as desired. In addition to ownCloud's desktop client, which is available for PCs running Windows, MacOS or Linux, ownCloud offers iOS and Android clients to cover most mainstream mobile operating systems.

All clients have the same technical structure in both editions, but as part of ownCloud's dual licensing strategy the licenses they are available under can differ. Both clients are split into a front end and a logical component. The frontend handles the display of the data for the user and was developed by ownCloud Inc., whereas the backend (logical component) enables the synchronization mechanism and was in-licensed under MIT licenses in accordance with ownCloud's policy of only reusing components under a permissive license or LGPL.

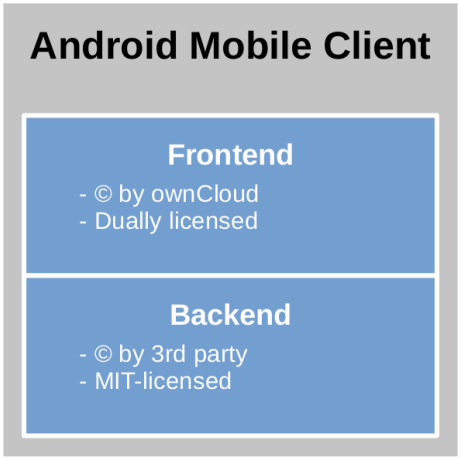

Figure 5: Setup of the Android mobile client

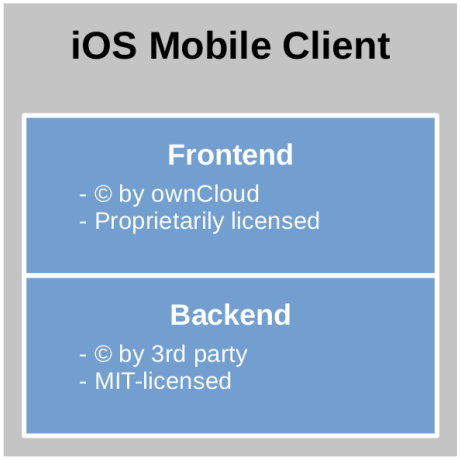

Figure 6: Setup of the iOS mobile client

Originally, the development of the clients had been undertaken by ownCloud Inc. In August 2012, when ownCloud released the first version of its mobile clients, it was decided that only the Android client was to be open-sourced in order to enable the community to participate. In line with the dual licensing strategy, the Android client was out-licensed via a GPLv3 for the Server Edition and under an ownCloud proprietary license agreement for the Enterprise Edition. With regard to the iOS client, the ownCloud management decided at that time that they would only be able to out-license it under a proprietary license agreement.

In 2010 Apple, for its part, had stopped accepting any apps licensed under a GPL license as it was not possible to rule out incompatibilities between the license and the iOS App Store terms (Contributions, 2010). But ownCloud's distribution strategy of mobile clients — as designed by the three co-founders — differentiated between the Server and Enterprise Edition. Users of the Server Edition would be able to purchase clients from the online app marketplaces of Ap-

ple and Google only. The right to redistribute clients freely to their end-users as needed would be retained exclusively by ownCloud's Enterprise Edition customers. To keep that strategy intact, the iOS client of the Server Edition had to be compatible with the terms of the iOS App Store. Thus the co-founders came to the conclusion that they would need to go without an open-sourced version of the iOS client for the present.

4.2 The licensing of the iOS client

Holger Dyroff, still waiting at the traffic light, recalled the situation in which their decision, made back in 2012, had left them. He now reviewed the differences between the Android and the iOS client.

The Android client was dual-licensed and available under the GPLv2 and a proprietary license agreement, whereas the iOS client was available only under ownCloud's proprietary license agreement. Also, the source code of the iOS client was not made publicly available. ownCloud Server Edition users and community members had no choice but to use this proprietary application if they wanted to get the best ownCloud experience on their iPhone or iPad. Android users, however, did not have to leave the FLOSS universe as they were offered the GPLv2 licensed client. Consequently, only Android users were able to review the code of the client and to participate in the development of the software.

Moreover, the clients had been developed using different programming languages. The iOS client is based on Objective-C technology whereas the Android client employs Java, which is more popular in FLOSS communities (Metz, 2015).

There was a third difference in terms of the way in which the clients could be distributed. In order to be obtainable by users, the mobile clients had to be offered through the application marketplaces of Google and Apple. The Google Play Store accepts GPL-licensed applications for distribution. In principle, ownCloud Server users were able to upload their own variations of the ownCloud Android client, also in branded form even though ownCloud Inc. aims to ensure that only Enterprise Edition customers should be given the option of labeling clients with their own logo. That made the Android client more commercially utilizable for ownCloud Server users. Without the Enterprise Subscription, OEMs could also offer an Android sync client with their corporate logo without needing to subscribe to the ownCloud Enterprise edition. OEMs needed only to be willing to accept the GPL conditions and to open-source their code if required to do so by the license.

The ownCloud management team had a certain strategy in mind when the mobile clients were first launched. They agreed that the Android app would be more important for the community, as Android had more users in the ownCloud community than iOS did. Therefore, and in order to comply with their FLOSS principles, they did not hesitate to open-source the Android client. But ownCloud Inc. was also seeking to recruit larger businesses as customers for their Enterprise Edition. Apple devices were known to be very popular in that customer segment. The rationale of Dyroff, Karlitschek and Rex was that the customer group they were targeting would not be able to also distribute an iOS sync client in their portfolio in order to cover all requirements for an efficient EFFS solution.

Thus, the iOS client was offered solely under a proprietary license not only because of the incompatibility of GPL and Apple's iOS App Store. The iOS client was also assigned a proprietary license agreement to avoid giving the target group another reason to stick with the gratis ownCloud Server solution. In this way, it was ensured that no potential customer would be

able to offer the complete set of sync clients. By using this approach, the co-founders hoped to create one more incentive that would persuade potential customers to choose the Enterprise Edition. Also, OEMs would not have the opportunity to structure their own offer with their logo on iOS clients. So the strategy not to open-source the iOS client as the only component of the ownCloud architecture was adopted with an eye to generating revenue and hence commercial success — factors even the most dedicated FLOSS firm finds it hard to neglect.

But the strategic decision came at a price. It did not take long for a lively discussion to be sparked inside the community. Some members simply complained that they were not able to review the code, or requested open-sourcing of the code in the near future; ownCloud, they insisted, should also openly publish the iOS source code on GitHub.[18] Others even questioned the whole ownCloud solution, because it had a fully proprietary component. Although only a fraction of the large community was really troubled by the setup, the arguments weighed heavily in the minds of the three co-founders.

In addition, ownCloud management had to realize that the development of the Android application was progressing much faster than that of the iOS client. The iOS app was missing features which already had already been implemented in the Android solution. The iOS client was developed with the exclusion of public contributors. As there were no user stories of handling by the community that could be published, all progress needed to be made by ownCloud employees. Last but not least, the ownCloud iOS development team had no assistance with finding, characterizing, and fixing bugs.

Dyroff wanted to use the meeting with the other co-founders to finally agree on measures to address these issues. On the other hand he knew they would have to be careful and keep the right barriers in place in order to prevent potential customers from getting too cozy with their gratis FLOSS offering. Like any company, they could not allow their business success to be jeopardized. If they were to open-source the iOS client, which FLOSS license would it be best to use? Was a multiple licensing strategy, as with the other clients, ultimately the better way? Could they find a solution that would allow for both; that would eliminate the negative effects associated with the iOS client while also providing an incentive for potential customers to subscribe to the Enterprise Edition?

18 GitHub is a web-based repository hosting service that allows for the collaborative development of software. Since 2012, the activities of the ownCloud community have been undertaken on GitHub as a public repository (https://github.com/owncloud).

References

Arlotta, C. (2014, July 8). Gartner Magic Quadrant for Enterprise File Synchronization and Sharing. Retrieved February 20, 2016, from http://talkincloud.com/cloud-computing-research/070814/gartner-magic-quadrant-enterprise-file-synchronization-and-sharing

Contributions, B. (2010, May 26). More about the App Store GPL Enforcement. Retrieved January 16, 2016 from http://www.fsf.org/blogs/licensing/more-about-the-app-store-gpl-enforcement

Endsley, R. (2011, December 14). Former SUSE Exec Joins Open Source ownCloud, Launches Commercial Entity. CMS Wire, Retrieved October 25, 2015, from http://www.cmswire.com/cms/document-management/former-suse-exec-joins-open-source-owncloud-launches-commercial-entity-013849.php

Evans, D. S., & Layne-Farrar, A. (2004). Software patents and open source: the battle over intellectual property rights. Virginia Journal Of Law & Technology, 9, 10.

Fitzgerald, B. (2006). The transformation of open source software. Mis Quarterly, 587-598.

FSF (n.d.). What is Copyleft. Retrieved January 15, 2016, from http://www.gnu.org/copyleft/copyleft.html

Helmreich, M., & Riehle, D. (2012). Geschäftsrisiken und Governance von Open Source in Softwareprodukten. HMD Praxis Der Wirtschaftsinformatik, 49(1), 17-25.

Jaeger, T. & Metzger, A. (2011). Open Source Software: Rechtliche Rahmenbedingungen der Freien Software. München: C.H. Beck Verlag.

Karlitschek, F. (2014, June 19). Why I Built ownCloud and Made It Open Source.Linux.com, Retrieved November 10, 2015, from https://www.linux.com/news/enterprise/cloud-computing/777158-why-i-built-owncloud-on-open-source

Laurent, A. M. S. (2004). Understanding open source and free software licensing. Sebastopol, CA: O'Reilly Media.

Lerner, J., & Tirole, J. (2002). Some simple economics of open source. The journal of industrial economics, 50(2), 197-234.

Lindberg, V. (2008). Intellectual property and open source: a practical guide to protecting code. Sebastopol, CA: O'Reilly Media.

Metz, C. (2015, August 20). Github's Top Coding Languages Show Open Source Has Won. Wired, Retrieved November 03, 2015, from http://www.wired.com/2015/08/github-data-shows-changing-software-landscape/

IDC (2014). New IDC Worldwide File Synchronization and Sharing Forecast Shows Market Will Grow to $2.3 Billion by 2018. Retrieved October 25, 2015, from http://www.idc.com/getdoc.jsp?containerId=prUS25192614

Olson, M. (2005). Dual licensing. Open Sources 2.0, 71-90. Sebastopol, CA: O'Reilly Media.

ownCloud (2015). ownCloud kann Zahl der Kunden verdoppeln und Umsatz verdreifachen [Web Blog Post]. (2015, September 22). Retrieved November 10, 2015, from https://owncloud.com/de/owncloud-kann-zahl-der-kunden-verdoppeln-und-umsatz-verdreifachen/

ownCloud Logo [Image]. (n.d.). Retrieved February 19, 2016, from http://tsdr.uspto.gov/#caseNumber=85979290&caseType=SERIAL_NO&searchType=statusSearch

Raymond, E. S. (2001). The Cathedral & the Bazaar: Musings on linux and open source by an accidental revolutionary. Sebastopol, CA: O'Reilly Media.

Rex, M. (2016, January 26). ownCloud grows over 100% in 2015 [Web Blog Post]. Retrieved January 26, 2016, from https://owncloud.com/owncloud-grows-100-2015/

Riehle, D. (2012). The single-vendor commercial open course business model. Information Systems and e-Business Management, 10(1), 5-17.

Rosen, L. E. (2005). Open source licensing: Software freedom and intellectual property law. Upper Saddle River, NJ: Prentice Hall PTR.

Stallman, R. (n.d.). FLOSS and FOSS. Retrieved January 25, 2016 from http://www.gnu.org/philosophy/floss-and-foss.html

Stallman, R. (2009). Viewpoint Why open source misses the point of free software. Communications of the ACM, 52(6), 31-33.

Williams, S. (2002). Free as in Freedom: Richard Stallman's Crusade for Free Software.

Appendix

Exhibit 1

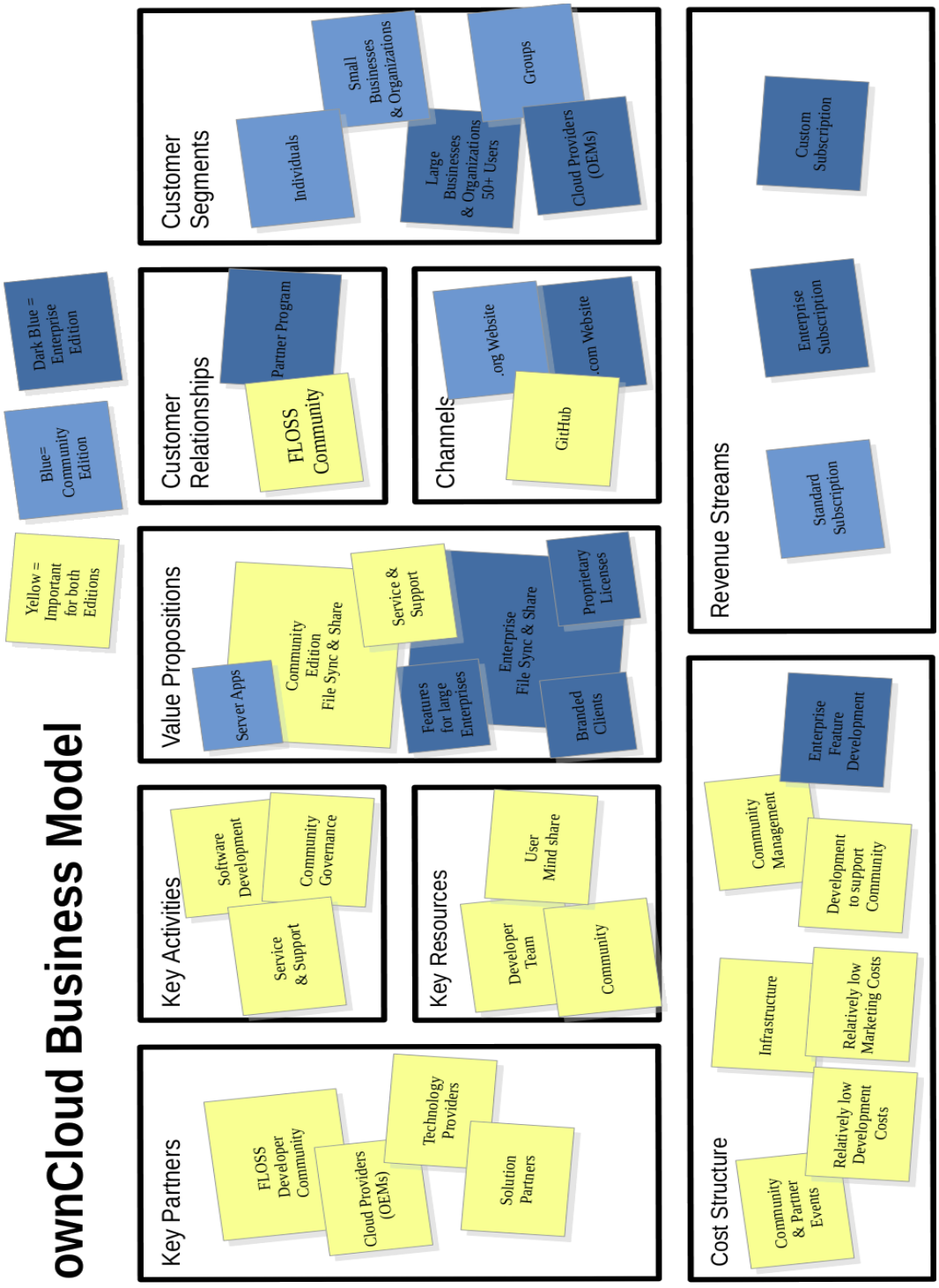

Exhibit 1: Summary of the ownCloud business model based on the the Business Model Canvas Template of Alexander Osterwalder

Exhibit 2

	ownCloud Server Edition	ownCloud Enterprise Edition
Type of Licensing	FLOSS-licensed	Proprietarily licensed
Gover-nance	ownCloud community	ownCloud Inc.
Targeting	FSS market • Individuals • Small- to medium-sized organiza-tions	EFFS market • Large organizations • OEMs
Based on		ownCloud Server Edition
Distribu-tion of mobile clients	Google Play and IOS App Store • Users need to purchase mobile clients from the online app market-places of Apple and Google.	Companies can redistribute clients under their own terms • Customers can obtain mobile clients directly • Companies can redistribute clients to their employees or own customers • End users do not have to purchase mobile clients from the Apple or Google online app markets
Additional value for users	Community Applications • Available for a fee via the ownCloud App Store • Offer additional features • ownCloud as a platform for function-ality beyond FSS • Sample apps: TextEditor, Anti-virus, Gallery, etc.	Enterprise Applications • Distributed in combination with the ownCloud Core • Features typically interesting for large companies • Sample apps: Microsoft Share Point Integration, Logging, File Firewall, etc.
		Commercial License • Enterprises can avoid the FLOSS-associated risk of having to unwill-ingly share • The Commercial License offers al-most the same freedoms as FLOSS licenses with the exception of redis-tribution
		Rebranding of Clients • Customers can use their own logo to integrate ownCloud into their corpo-rate design
Availabil-ity	Freely downloadable and usable • Support via the community forums	Enterprise Subscription • 5x12 email and phone support • From 50 up to 100,000 users
	Standard Subscription • 5x8 email support • From 50 up to 1,000 users	Custom Subscription • 24x7 email and phone support • From 10,000 users

Table 1: Comparison of ownCloud editions

Exhibit 3

	Derivative Work	Combined Work	Collective Work
Type of Modification/ Combination	Modification/ incorporation	Linking	Mere aggregation of single components
Example application	Modification of reciprocal licensed source code to add a new feature	Usage of libraries or plug-ins which are under a reciprocal license	Combination of different programs in a software suite with one component under a reciprocal license — similar to Linux distribution
Copyleft effect	Resultant software is distributed under same reciprocal license	Whole combination is distributed under same reciprocal license — exception: LGPL	No effect; no freedom of choice with regard to licensing options for any elements of combination— all components keep their original licenses
Likelihood of viral effect	Certain	High	Low
Compatibility	All other licenses need to match the terms of the reciprocal license used	All other licenses need to match the terms of the reciprocal license used	Unproblematic

Table 2: The various copyleft effects associated with different forms of modification and combination of software

Exhibit 4

License Name	License Type	Usage at ownCloud	Key Properties	Derivative Work	Combined Work	Collective Work
GPLv2	Reciprocal license	Out-licensing	Extensive copyleft effect Attribution	Derivative must be released under GPL	Combination must be released under GPL	No restrictions
GPLv3	Reciprocal license	Out-licensing	Extensive copyleft effect Enhanced compatibly Software patents Attribution	Derivative must be released under GPL	Combination must be released under GPL	No restrictions
AGPLv3	Reciprocal license	Out-licensing	Extensive copyleft effect Network reciprocity Software patents Attribution	Derivative must be released under AGPL	Combination must be released under AGPL	No restrictions
LGPL	Reciprocal license	In-licensing	Weak copyleft effect Attribution	Derivative must be LGPL- or GPL-licensed	No restrictions	No restrictions
MIT	Permissive license	In-licensing	Attribution	No restrictions	No restrictions	No restrictions
2-clause BSD	Permissive license	In-licensing	Attribution	No restrictions	No restrictions	No restrictions
Apache 2.0	Permissive license	In-licensing	Attribution Software patents	No restrictions	No restrictions	No restrictions
ownCloud commercial	Proprietary license	Out-licensing	No redistribution allowed Otherwise grants rights similar to those permitted by FLOSS			
ownCloud EULA	Proprietary license	Out-licensing	No rights granted — no free use, no open source code, no modification, no free distribution — unless explicitly permitted by ownCloud			

Table 3: Most important licenses used by ownCloud

About this case

This case was taken from the Product Management by Case collection, a collection of free business cases for teaching product management, available at http://pmbycase.com.

Conceptual guidance and teaching notes are available to lecturers. To receive these, please send an e-mail to to dirk@riehle.org.

Case license

Case authors

Content contributions by Johannes Christian Neupert, Dirk Riehle.

Copy-editing by Grace Ting and FAU Sprachendienst.

Case data

CaseShortCode: Case-2016-01-Getting-Licensing-Right

CaseFirstAuthor: Johannes Christian Neupert

CaseFileLicense: CC BY SA 4.0

CaseRevision: 180807

More cases from PM by case

Case 2012-01: Ensuring innovation at Method Park

Case 2013-02: Two-sided markets at Netdosis

Case 2014-01: User experience design at Immowelt

Case 2014-02: Switching suppliers at Nokia

Case 2014-03: Specifying ‘wow!’ at Elektrobit

Case 2016-01: Licensing choices at ownCloud

Case 2016-02: Stock options at Caldera

Case 2016-03: Hard software marketing choices at ownCloud

Case 2016-04: Pricing at Everest SARL

Case 2017-01: The case of SUSE Manager