Dual Licensing in Open Source Software Industry

Johannes Lauri, Janne Leimurautio & Ari Vuollet

Abstract

Open source and licensing models are important part of some companies in software business. Licensing model will provide to companies the opportunity to exploit open source's benefits such as a wide developer communities. One principle of the open source is that people have the freedom to use software for any purpose, copy and distribute both, original and edited version. The problem is, how to make money with open source? One answer for this questions is dual licensing, which offers two versions of product. One version can be licensed as, for example, GNU General Public License, and another version of the product is proprietary version. Dual licensing offers users more choices how they may use the given software in conjunction with their own projects, providing tailored licenses for both open source and other proprietary developers. Dual licensing options usually ranges between proprietary and strong copyleft licenses for the same piece of software. This in turn allows the companies to monetize their product with proprietary license and additional services alongside their open source alternative. Unfortunately, there has been too few studies about licensing of open source and especially about dual licensing.

Our purpose in this paper is to find out what are the benefits and strengths of dual licensing, especially in software business industry. We will produce a literature review and try out to solve our problem by looking existing studies from researchers and comments which are made by the experts of the field.

We found out that there are several benefits that dual licensing offers. For example it can lure out a wider developer community and decrease marketing costs. It can satisfy two different community actor with the two different license. One weakness of dual licensing is its ability to create and maintain codebase that has the ability to be a dual license. Also because the trend is now to use permissive licenses, competition with the commercial projects are harder with the dual licensing model.

Dual licensing have some important benefits why it is, or at least has been, middling licensing model. However, it has been declining in recent years and this can be the result of the challenges what it reveals and the increased awareness of open source.
.

1. Introduction

Most obvious reasons for companies not to open up the source code of their products with open source licences are economical. After source code has been released anyone can modify and use their product in whatever way the license grants them. This usually takes away the possibility to profit from the software by selling licences to the users of the software. Many different models for making profit after releasing the source code have been used. For example companies might end up selling services around their product.

One possibility is to offer the source code with two or more licences. This practice is called dual licensing. In dual licensing-model software vendor provides their software with open source and proprietary licence. Customers who are willing to live with restrictions of open source licence can do so, but there is also an option to acquire proprietary version of the source code. Proprietary licence allows customers to acquire source code, and build their products on top of the software without the need of publishing their modifications and additions to source code.

Recent survey of 114 software vendors showed that use of dual licensing had decreased between 2008 and 2010 from 16% to just 5%. (Aslett, 2010) In this literature review we try to find answers to following research questions related to declining use of dual licensing.
What are the weaknesses of dual licensing model? What kind of benefits dual licensing model has?

2. Dual Licensing

Dual licensing, or sometimes referred as multi-licensing or proprietary relicensing, is a licensing scheme where open source version of the software is available under the usual open source terms, while proprietary version is available for a fee (Fogel, 2005). In this model, the customer can choose the license under the terms of open source licence, such as the GNU General Public License (GPL). Alternatively, customer can choose a proprietary license that offers more

typical licensing terms that restrict the customer's possibility to re-license the product (Meeker, 2005.)

Dual licensing model differs from the completely free software products in many ways. Big difference in projects utilizing dual licensing is that instead of being community driven, dual-licensing projects are organized by one vendor who controls the projects. Dirk Riehle (2010) calls this type of open source project *single-vendor commercial open source*. In single-vendor commercial open source it is crucial that vendor holds full control over the project, by owning the full copyright to the code and related intellectual property such as patents and trademarks (Riehle, 2010.) Development community can't start new competing products, because copyright is in the possession of the original developer. Thus, it means that it requires a full ownership to all rights if the product is wanted to license by different terms than open source (Välimäki, 2005.)

Another way how dual licensing model differs from free software product models is that those users, who are using free license, get option to obtain proprietary license. Combined product, which has reciprocity obligation and which is embedded to come as another part of another product, should be shared for free. Getting rid of this restriction is possible with help of proprietary license and therefore third party product businesses are possible (Välimäki, 2005.) Välimäki (2005) writes that we can come to conclusion from previous difference that dual licensing is not discriminating if we think from the user perspective.

Watson (2008) names firms using dual-licensing model as a second generation OSS companies. They can be described as a hybrid model of corporate and open source models. For these companies, most of the generated revenue comes from selling proprietary licenses to the customers who have specific needs for it and are not satisfied with open-source licenses (Välimäki, 2003). In addition to this, the companies may also sell support and other services around their products. There are several companies that are famous for providing dual licensed products. Few to mention: MySQL is perhaps most used company in our references. Blanco (2012) also named Qt from Nokia (originally Trolltech), Berkeley DB from Oracle (originally Sleepycat) and from Digium he mentioned Asterisk.

2.1 Benefits of dual licensing

Software developers and vendors will benefit from the open source community work if they release an open source version of their product for free. This work may added to the proprietary version and by this way they can improve their product's quality (as cited in Comino and Manenti, 2010).

Aslett (2008) names three strengths and benefits that dual licensing have especially in by users point of view. The first is that community-oriented version and commercially-oriented version

has a clear difference. Second strength is that free version community users are happy with that there is no exploitation from features and functionalities and the same time commercial proprietary version users are happy that they will from commercial relationships all the benefits. As third strength is mentioned copyright controls (Aslett, 2008.) It also gives lower cost, better reliability and more viable relationship with the vendor to the users (Wynants & Cornelis, 2005).

Former president and CEO of Sleepycat, Michael Olson, told that moving to dual licensing strategy was a beneficial because it allowed them with the same product to share open source version and the same time earn from software license. Open source license brought them a huge amount of developers (Winterspeak, 2001.) Dual licensing can save also the marketing costs and the reason for this is that the product can get well known even without a big marketing budget (Wynants & Cornelis, 2005). For example many vendors try to make their products known in the next generation of software professionals by offering special deals for universities. Dual licensing offers very simple model for using that software in education and research for free, because there's no need for special low- or no-cost licenses. (Olson, 2005.) The fact that they used a dual license strategy helped them to recruit developers and, among other things, marketing staff. This led forward that they were able to promote Berkeley DB (Winterspeak, 2001). Another good example is case of MySQL. Makers of closed programs will get good tools in their hands for the quite good price and the same time open source developers will get these same programs for free, by certain terms which they seemed accept well (OpenLife, 2012.)

Another benefit of dual licensing that Olson (2005) mentions is that it might be really hard for companies to influence community driven open source projects towards their needs. But when project is driven and controlled by single-vendor, paying customer can have much more influence in steering the project to certain direction, and to get attention to features that are important to them (Olson, 2005.)

2.2 Challenges for dual licensing model

Thinking from the point of view of industry, some companies may want to avoid open source and do not want to invest or buy a companies like for example start-up company who is playing in the open source field (Meeker, 2005). One of the most long-term, popular and succeeded dual licensing user company is MySQL AB (Välimäki, 2005). Difficulties and challenges which MySQL AB had to face in their dual licensing model was for example that when MySQL and PHP made changes to their licenses. These changes brought up conflict, because licenses were not suited well with together when PHP was combined with MySQL (Gomulkiewicz, 2004.) Dual licensing points out also three concerns to the customer: availability of support, vendor accountability and vendor viability (Wynants & Cornelis, 2005). Implementation of the dual license does not, however, always improve the user satisfaction or welfare (Darmon & Torre, 2014). Brian Aker said in year 2009 that important area of dual licensing is the ownership rights. These rights will lead to into clash with the principles of open source and developers who are

contributing the code. By the effect of dual licensing any developers who want to contribute will have to leave their rights and agree to that their work could end up in commercial software (Aker, 2009.)

2.2.1 Negative effect on community contributions

Dual licensing requires copyrights of the contributions to be assigned to the vendor for example via Community License Agreements. Many authors argue that this can have negative effect on community contributions to the project. (Aslett, 2008; Blanco, 2012; Hillesley, 2011) According to Carlo Daffara (2009) requirement to accept this licensing regime has been shown to decrease the volume of external contributions. Hillesley (2011) mentions that Community License Agreements can strip away the clarity of a project's purpose and this can keep people from contributing to the project. Keeping both user groups happy in the process can be challenging for the vendor, because features demanded by another group might be different from the needs of the other group. Unhappy community might prevent vendor from enjoying the full benefits of the open source projects (Aslett, 2008.)

2.2.2 Improved understanding of open source licenses

In his article Simon Phillips (2012) argues that use of dual licensing might be declining because today GPL and its effects on existing codebase are understood better. He suggests that reason why companies originally bought the commercial versions of dual licensed software was fear of contaminating their own codebase. To play it safe companies would get the commercial license just of out of fear of GPL making them to open up their internal systems to the world. (Phillips, 2012) Thomas Prowse (2010), who has worked as Global Law Department leader in Nortel Open Source Advisory Team with OSS matters, also believed that rise of education and sophistication of end users toward OSS licensing is the one reason why dual licensing is declined. Prowse also said that dual licensing has some challenges to create and maintain codebase that has capability to be dual licensing (Prowse, 2010).

A trend towards more permissive licenses in open source industry has been noticed. More projects are using permissive licenses, and companies prefer to use open source software which is licensed with permissive license. Strong copyleft licenses like GPL require extra work from both legal and development teams to find ways to work with it in commercial projects. (Asay, 2013)

2.2.3 Types of software

Dual licensed projects have to be constructed so that users need to combine it with their software. If the user wishes to embed dual licensed software within their products, in case of GPL this requires their existing licenses related to the project to be compatible with GPL, otherwise they need to change their proprietary license or consider other options (Widenius and Nyman, 2014.). Combination with the reciprocal licence forces users to license their work with compatible license (Olson, 2005). Dual licensing model is only effective when licensed product is meant to be embedded inside another product (Widenius and Nyman, 2014). Most successful dual licensed products are meant to be embedded inside of another product (Onetti and Verma, 2009).

2.3 When to use dual licensing model

For dual licensing to work, software must be licensed using strong copyleft license like GPL, and all contributors must sign their copyrights to the software provider. Dual licensing model works because copyright owner has a right to choose whether he wants to sue himself over a copyright infringement or not (Fogel, 2005.)

There has been studies where is mentioned some factors when to use dual licensing model. Some studies reflect indirectly to what is the best situation to use dual licensing model but in this section we will present some indirect situation and direct situations when to use dual licensing model and we also reflect to some example case studies by Välimäki (2003) where a couple of popular companies started to use dual licensing as their open source business model. Mika Välimäki (2005) is telling in his book that dual licensing can be commercially viable if there is a demand for stand-alone or/and for embedded products (Välimäki, 2005). In particular, dual license can be considered to be best open source business model expressly for products that are embedded and where one owns the code. This way the primary customers, who are typically companies, and who want to include software to their packages, but are not willing to share their code under open source, prefer dual licensing model (Widenius & Nyman, 2014).

Välimäki (2003) studied three open source company's dual license using and made from them a detailed analysis. Companies were selected by the fact that at the time there were just a few companies which was known to be using dual licensing model successfully (Välimäki, 2003).

First company what Välimäki studied is Sleepycat. Sleepycat was a company which maintained Berkeley DB packages from 1996 all the way up to 2006 when Oracle Corporation acquired it (InformationWeek, 2006). Before the actual foundation of the company, the product itself began to receive more attention from users and it also received commercial interest. After the establishment of the company, they developed product further and later this product was even more commercially valuable. It was released under the Sleepycat License in 1997 because they wanted that open source community would use their library but in the same time they wanted to make money, and since then, the product has been under dual licensing model (Winterspeak, 2001; Välimäki, 2003).

Second company of Välimäki's study is MySQL AB which has been mentioned in this paper earlier. Development of MySQL started 1995 by act of Michael Widenius and David Axmark. MySQL delivered at first his own license and therefore distribution was free but limited and using product in UNIX systems was possible. However, license model in windows was shareware which restricted free use and distribution of product. They were quite close dual licensing with their UNIX systems and in windows product was proprietary. When MySQL started to become popular in Linux, free license was changed to GNU GPL during year 2000. This caused that their license model was purely dual licensing. Because of this license change they got a lot more users to their product (As cited in Välimäki, 2003.)

Qt, which is the product of TrollTech and Välimäki's (2003) third example in his study, was started to develop during year 1992. When Qt was released in 96, it was released under restricted open source license. This license didn't allow distributing modifications freely (Välimäki, 2003). Eirik Eng told in his interview in KDE website (2001) that they changed their license to GPL even though they feared that someone will come and make a hostile fork and take over their only product. They already had some increasing rate of growth before the change. Qt's open source code made it possible for KDE, open source community and desktop environment (KDE, 2015.), to start using it. KDE became a very popular in UNIX and Linux systems (Välimäki, 2003.) Erik state that KDE was one significant factor in their success. Because of KDE, many customers got to know Qt and if engineering developers liked Qt, they could asked for their bosses to buy it (KDE, 2001.)

As we can find out from the previous examples: Using dual licensing can be a good choice when company want to get more users to use it product and in the same time make money with the product. Previously mentioned companies have used dual licensing a quite well. However, according to OpenLife website (2012), for all companies this model is not suitable like for example for companies that has done their product for just to end users. This is because usually the users of this model sell their products, like libraries or databases, to programmers (OpenLife, 2012).

For dual licensing to work the software product must be so attractive that users simply have to use it. The software product must create a need in the market. Software must be constructed so that users must combine it with their own intellectual property. At the same time open source license must make this activity painful to at least some of the customers, so that some users would rather pay money than endure the pain. On the other hand, to make large market penetration possible, the open source license must not be painful to the open source community. (Olson, 2005)

3. Discussion

Historically dual licensing was one of the first ways to make money with open source.

Dual licensed project is not a regular open source project, it’s a commercial software project which source code is open and available for free over restrictive copyleft license, and for a fee over permissive proprietary license. For dual licensing to be effective in creating revenue streams the licensed software must be embedded inside of another product. Dual licensing doesn't work with software that is targeted towards end-users. Because of this dual licensing is used for software products that people use to build their products on top of.

In the past few decades commoditization has been happening in computer industry, starting from hardware and working its way up to ever higher levels of software stack. With successful projects like Linux, open source movement has had a big impact on commoditization of software industry. (Murdock, 2005). During the past few years web companies like Facebook, Google, Twitter and others have started to open up their platforms. These companies are looking for larger developer adoption for their software platforms rather than profit, for them the technology itself is not the differentiator, but ability to operate that software in large scale (Asay, 2013.)

Commoditization of software platforms can be viewed as a threat for dual licensing model. Dual licensing only works with certain types of software products, and there seems to be a trend to open up the software platforms with permissive licenses.There is some evidence that certain elements of dual licensing scheme might have negative effect on community contributions. In this light more permissive projects, even if they are still single vendor driven can be more attractive to open source communities.

Same survey which found that use of dual licensing is declining, also noticed increase in use of open core licensing model (Aslett, 2010). Similar to dual licensing business model, open core offers the core product under multiple licenses, but in addition to the product additional features are provided under proprietary licenses. These vendor exclusive extensions are usually considered very vital to the core product, luring users to switch over to proprietary licensing to gain access to these features not available for open source community. This model has received some criticism and has been described as not a true OSS business model, forcing users to vendor lock-in along with their provided services and non-GPL-compliant licenses (Hillesley, 2011) Some projects that used to use dual-licensing have moved at least partly to use open core model. Most well known such project is probably MySQL (Widenius and Nyman 2014).

While from the open source viewpoint dual licensing seems to be better option because there is no difference between the free community version and the paid version as Aslett (2008) had noticed. Rising adaptation rates gives indication that companies might prefer to license their products with open core model over the dual licensing model. Open core is in some ways step back to the more closed source development.

Today large number of software is written to be used as a service over network connection. General Public License (GPL) does not require release of modifications if the programs are not distributed outside of the organization (Free Software Foundation, 2015). This gives cloud software vendors loophole from having to contribute the changes they made to the software, since cloud software resides on private servers and is not distributed to it's users. In the industry this loophole is called Application Service Provider (ASP) loophole, or SaaS loophole. In basic terms it means that dual-licensing is not effective model in cloud software if that software uses GPL-style copyleft license. This is a major risk if organizations funding model is based on dual-licensing and somehow someone is able to run your software as a service on a remote server.

Affero GPL (AGPL) was created to fill ASP loophole that existed in the original GPL license by adding a clause that required modifications to be shared to users if they were interacting with it remotely through computer network (Free Software Foundation, 2007). AGPL has not been very widely adopted in open source projects. For example Google has completely banned any software using AGPL, due to the complexity of code sharing requirements of the license and engineering time it would require to keep some parts of the codebase proprietary (Metz, 2011).

ASP loophole, low acceptance rate of AGPL, and commoditization of software platforms especially in web domain can make it hard to have financially successful web oriented dual licensed products. It might be the case as some people have argued that companies might have acquired the proprietary license just out of the fear of GPL, but today special cases like ASP loophole are understood better and companies can find ways how to work around limitations of copyleft licenses. This makes selling licenses harder, and we suspect might be one of the reasons why some companies are adapting models like open core, which allow companies to have protected revenue sources.

Trend towards more permissive licenses in both projects and inside companies, makes it harder to compete with dual licensing model. It is quite challenging to use dual licensed software in your own commercial project without acquiring the proprietary license, or at least you had to make sure that value of your software product is somewhere else than in its source code. If there is no market for certain type of commercial closed source project, it is very unlikely that market for dual licensed open source project exists either.

When dual licensed projects are compared to proprietary projects, they have many benefits and can attract larger groups of contributors and users. For example attracting new user groups is easier, because there is no need for special low-cost or trial plans since software is available for everyone to use. But when dual licensed projects are compared to more permissive community driven open source projects some benefits of open source development model are lost. In some cases dual licensing holds advantages over community driven open source projects. For example it might be easier to steer the project and influence the development priorities for paying customers in dual licensed projects than it would be to try to influence the open source

communities. Possible risk is that needs of the community and paying customers are in conflict with each other, and some development choices can anger the community users.

4. Conclusions

The aim of this article was to find out what are the weaknesses of the dual license, as well as what are its strengths, for example, for companies. Our research method was to make a literature review where we included online writings of professionals with regular literature.

Challenges and weaknesses that dual licensing face, are partly the result of an open source license, proprietary license, as well as their use with together. In the past, companies have been a little leery with the open source, but nowadays issue can be seen in such a way that there has been positive progress in the minds of companies. Awareness of open source has also brought challenges to dual license. Earlier companies bought the proprietary version of dual licensed software to avoid that their codebase would be "defiled". The weakness of dual licensing is its ability to create and maintain codebase that has the ability to be a dual license. The trend is now that companies want to use more permissive licenses in both, projects and inside the company, thus the competition with more permissive open source projects with products using dual licensing model is harder. Big challenge for dual licensing is that it only works with certain type of software projects.It is not suitable for end user applications and has to be used in a product that is meant to be embedded inside another product.

We can see also clear benefits what dual licensing can bring. In general, proprietary version and the open version has a clear difference that makes the dual licensing a reasonable solution. Each version users will benefit from their qualities and trust toward vendor can be better. It can also lower the marketing cost because this licensing model can give attention to software to be more known in the field. One of the biggest benefits that dual licensing can be said to bring out, is the fact that it can provide a large number of developers, either from the community or hired developers (which may be possible by money received from proprietary version).

Despite the challenges, many organization has been able to include dual licensing as their business model. Good example of such company is MySQL. However, many projects have moved at least in some level to use open core model or some other monetization policy. It can be in a someway stated that dual licensing is reducing, we think that mostly because of the challenges we proposed in this paper but also because the awareness of education and understanding of the OSS licenses has increased in recent years.

For further studies we recommend a more comprehensive research about why use of dual licensing is declining. Also, so far there is just little knowledge in research field about dual licensing, and licensing models as whole, so it would be also good if some effort will be added to this domain.

References

Aker, B (2009). *RMS, GPL, The Peculiar Institution of Dual Licensing* in LiveJournal.com. Retrieved from http://krow.livejournal.com/673195.html

Asay, M. (2013). *Is open source sustainable?* Retrieved from http://timreview.ca/article/650.

Aslett, M. (2008). *Open source is not a business model*. Retrieved from http://download.microsoft.com/download/0/4/2/04246FB1-0BF6-44F4-BC44-4CCB220E1711/451_-_08_OCT_-_CAOS_Report_9_Open_Source_is_Not_a_Business_Model.pdf

Aslett, M (2010). *Winning and losing with open core*. Retrieved from https://blogs.the451group.com/opensource/2010/03/25/winning-and-losing-with-open-core/

Blanco, E (2012). *Dual-Licensing as a business model*. Retrieved from http://oss-watch.ac.uk/resources/duallicence2

Comino, S., & Manenti, F. M. (2010). Dual licensing in open source software markets. *Information Economics and Policy, 23*(3), 234-242.

Darmon, E., & Torre, D. (2014). Open source, dual licensing and software competition. *Dual Licensing and Software Competition (March 2, 2014)*.

Fogel, K. (2005). *Producing Open Source Software: How to Run a Successful Free Software Project.* O'Reilly Media, Inc.

Free Software Foundation (2007). *Gnu Affero General Public License*. Retrieved from https://www.gnu.org/licenses/agpl.txt

Free Software Foundation (2015). *Frequently Asked Questions about the GNU Licenses.* Retrieved from http://www.gnu.org/licenses/gpl-faq.html#GPLRequireSourcePostedPublic

Gomulkiewicz, R. W. (2004). Entrepreneurial Open Source Software Hackers: MySQL and Its Dual Licensing. *Computer L. Rev. & Tech. J.*, *9*, 203.

Hillesley, R. (2011). H-online. *Open core or dual licensing? The example of MySQL*. Retrieved from http://www.h-online.com/open/features/Open-core-or-dual-licensing-The-example-of-MySQL-1367824.html

InformationWeek (2006). *Oracle Buys Sleepycat, Is JBoss Next?* Retrieved from http://www.informationweek.com/oracle-buys-sleepycat-is-jboss-next/d/d-id/1040539?

KDE (2001). *Interview: Trolltech's President Eirik Eng*. Retrieved from https://dot.kde.org/2001/09/24/interview-trolltechs-president-eirik-eng

Meeker, H. (2005). *Dual-Licensing Open Source Business Models.* Retrieved from *http://linux.sys-con.com/read/49061.htm*

Metz, C (2011). *Google Open Source Guru: 'Why we ban the AGPL'. The Register.* Retrieved from http://www.theregister.co.uk/2011/03/31/google_on_open_source_licenses/

Murdock, I. (2005). *Open Sources 2.0, Chapter 6: Open source and the commoditization of software*. Retrieved from http://commons.oreilly.com/wiki/index.php/Open_Sources_2

Olson, M. (2005). *Open Sources 2.0, Chapter 5: Dual Licensing.* Retrieved from http://commons.oreilly.com/wiki/index.php/Open_Sources_2

OpenLife (2012). *Dual licensing (MySQL, Trolltech Qt)*. Retrieved from *http://openlife.cc/onlinebook/dual-licensing-mysql-trolltech-qt*

Phillips, S. (2012). *What's next after GPL and Apache?*. Retrieved from http://www.infoworld.com/article/2616665/open-source-software/what-s-next-after-gpl-and-apache-.html

Prowse, T. (2010). Q&A. *What business models are currently used with open source software?.* Open Source Business Resource. Retrieved from http://timreview.ca/article/366

Riehle, D (2010) *The Single-Vendor Commercial Open Source Business Model*. Retrieved from http://dirkriehle.com/publications/2009-2/the-commercial-open-source-business-model/

Välimaki, M. (2003). Dual licensing in open source software industry.*Systemes d'Information et Management, 8*(1), 63-75. Retrieved from *http://www.valimaki.com/org/dual_licensing.pdf*

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

Watson, R. T., Boudreau, M. C., York, P. T., Greiner, M. E., & Wynn Jr, D. (2008). The business of open source. *Communications of the ACM*, *51*(4), 41-46.

Widenius, M., & Nyman, L. (2014). The Business of Open Source Software: A Primer. *Technology Innovation Management Review.* Retrieved from http://timreview.ca/article/756

Winterspeak (2001). *Interview with Sleepycat President and CEO, Michael Olson*. Retrieved from http://www.winterspeak.com/2001/10/interview-with-sleepycat-president-and.html

Wynants, M., & Cornelis, J. (2005). How open is the future? (p. 481). Brussels: Brussels University Press