This article was published in Systemes d´Information et Management 1/2003

Dual Licensing in Open Source Software Industry

Author: Mikko Välimäki[1]

Affiliation: Researcher at the Helsinki Institute for Information Technology

Address: P.O. Box 9800, FIN-02015 HUT, Finland

Email: mikko.valimaki@hiit.fi

Tel: +358-50-5980498

Fax: +358-9-6949768

WWW: http://www.valimaki.org/

CV: Mr. Mikko Välimäki, LL.M, works as a researcher in the Digital Economy Program within Helsinki Institute for Information Technology, a joint research institute of University of Helsinki and Helsinki University of Technology, Finland. He is currently writing his PhD thesis, which will discuss licensing models of software firms. His main academic interests are in the areas of computer law and law & economics

Support: The author wishes to thank his colleagues at the Helsinki Institute for Information Technology as well as the Finnish National Technology Agency and company partners for the generous funding of the MobileIPR research project.

[1] The author wishes to thank Michael Olsen of Sleepycat Software Inc., Tonje Sund of TrollTech AS and Mårten Mickos of MySQL AB for kindly providing information on their companies and the participants of Free Software Business mailing list fsb@crynwr.com for fruitful email exchange discussing an earlier version of this paper.

This article was published in Systemes d´Information et Management 1/2003

Dual Licensing in Open Source Software Industry

Abstract. This paper analyses how several open source companies use dual licensing: both open source and proprietary licenses for one product. Three case studies based on the experiences of companies Sleepycat Software Inc., MySQL AB, and TrollTech AS illustrate the issue. Especially the legal and economic requirements of dual licensing are identified.

Keywords. Open Source, licensing, business model, copyright, software economics

This article was published in Systemes d´Information et Management 1/2003

1 Introduction

This paper introduces and explains a sustainable open source business model. Open source can be described as a new way of doing business in software industry. However, how to maintain sustainable sources of revenue while using an open source development and licensing model has been a well debated issue (Raymond, 2001, cf. Microsoft, 2003). Many start-up software companies that based their business model on selling add-on services to free products have failed.

Dual licensing is based on the idea of simultaneous use of both open source and proprietary licenses. The concept is still quite novel in the literature and even in informal discussions with open source scholars and practitioners (cf. Fink, 2002, p.40-). For example Raymond's widely cited paper focuses only on means to make "indirect sale value" with free software (Raymond, 2001, p. 134-). This would consist of add-on services and bundled products. In contrast, dual licensing model does not nullify the direct sale value of the software product.

Also Feller and Fitzgerald skip dual licensing. They first divide open source companies into two categories: pure-play (solely open source business model) and hybrid (open source and proprietary models mixed). Then they identify e.g. Sleepycat Software Inc as a pure-play open source company but go on confusingly to maintain that Sleepycat's business model would be based on selling support services (Feller and Fitzgerald, 2002, p. 47-48). In reality, Speepycat uses dual licensing and their income depends heavily on the sale of proprietary licenses (Sleepycat, 2003). This article maintains that companies using dual licensing are both pure-play and use a hybrid business model.

We begin with explaining how dual licensing works as a software business model. Then follows a more practical analysis based on the licensing policies and experiences of three open source companies: Sleepycat Software Inc. from the United States, MySQL AB from Sweden, and TrollTech AS from Norway. Finally, several legal and economic requirements for a successful use of dual licensing are identified.

Before we start, however, a note on terminology is in place. Throughout the text, terms open source and free software as well as commercial and proprietary licenses are used interchangeably. To be precise, terms open source and free software refer to separate social movements with different goals (Stallman 2002). In this paper, however, we are especially interested in the technical facts of open source code and free availability of the software. The societal implication of open source and free software including community development are only shortly mentioned.

2 Dual Licensing as Business

In this section we first discuss different generic software business models and then describe the dual licensing model. Our aim is to understand how dual licensing works as a business model.

2.1 Generic Software Business Models

Software business models can be distinguished from several perspectives, for example, depending on whether the software is sold as a product or service, structure of the sales channel, and income sources. (Rajala et al, 2001). Messerschmit and Szyperski emphasize the

difference between technical distribution channel (pre-installed, self-provided) and pricing method (subscription, pay-per-use, cross-subsidy) and conclude that users have four ways to acquire software: make, buy, license or subscribe (Messerscmit and Szyperski, 2001).

Perhaps the most common way of software companies to do business is to sell *software projects.* In this model, a software company sells its programming work as a service rather than the sole software. Typically, the software is sold or licensed to the user. As a business model, project business is not far from a taxi firm with limited scalability. More cars running (programmers) mean more money to be charged.

Next, the traditional model for software product business could be described as *software publishing*. In this model, the software is licensed as if it were sold as a physical product. Software publishing works in a somewhat similar way to print publishers who sell physical books commoditized from manuscripts.

The Internet as usage environment and distribution channel has enabled several new ways to do software business. *Software subscription* can be seen as a combination of the two traditional models. Commonly called as application service providing (ASP), subscription is a more interactive way to sell software as an online product with add-on services tailored to the customer. It works like any subscription service: pay your monthly fee or your line is cut.

Finally, different *free software* business models have emerged. Here, the core product is usually free to use and distribute. It is also often required that no user can charge for the use or distribution of the product (standard copyleft clause) or even any of its derivatives (inheriting or strong copyleft clause). Therefore, sales are based on indirect means that

leverage the potentially large and dynamic user base. Add-on services, bundled products and branding are essential indirect revenue sources.

Table 1 identifies four generic software business models.

	Software projects	**Software publishing**
Product focus	Customer project	Product family
Copyright	Licensed or transferred	Licensed with restrictions
Income	One-time project fees	License fees

	Software subscription	**Free software**
Product focus	Parametrized product	Core product
Copyright	Licensed with restrictions	Licensed with an open source license
Income	Service fees and application rents	Indirect from services, bundling, branding

Table 1. Generic software business models (edited from Rajala et al. 2001)

2.2 Dual Licensing Model

Dual licensing mixes several of the introduced generic software business models. Duality means that both the free software distribution mechanism and traditional software product business are combined. There is technically only one core product but two licenses: one for free distribution and free use and another for other (proprietary).

Dual licensing model differs from pure free software model in several ways. First, the development community does not have development power to start competing products (forks). Copyright and control of the core product development is held in one hand, the original developer. The ability to license the product with other terms than open source requires full ownership of all rights to the product.

Second, the users of the free license have an option to obtain a proprietary license. If a software product with an inheriting copyleft clause – as for example term 2b) in the GNU GPL License (GNU, 1991) – is embedded to become a part of another product then the combined product should be distributed for free. A proprietary license may free the user from this restriction. In this way, third party product businesses become also possible. From the user's perspective, dual licensing can be described as indiscriminating.

Figure 1 describes the dual licensing model in more detail.

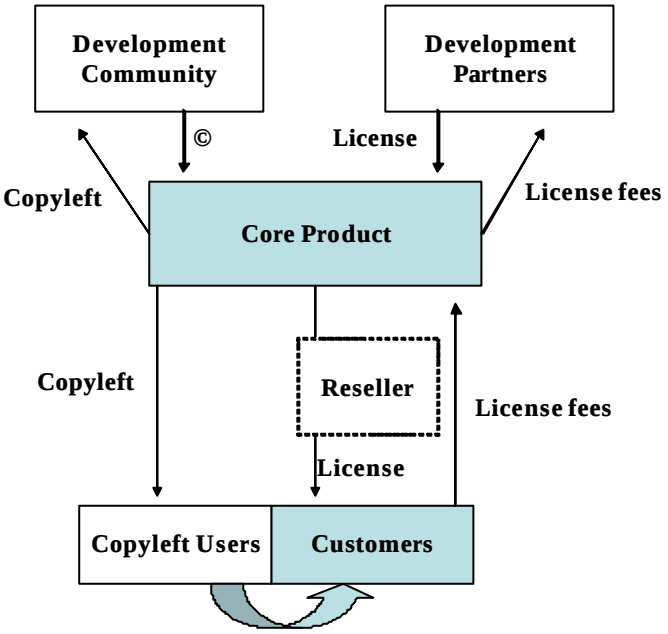

Figure 1. License streams of a core product in a simplified dual licensing model.

Let´s look at the figure 1 from the bottom up. At the bottom we have software users divided into two segments. They interact with the software company that dual licenses its *core product* with a strong copyleft license to the first user segment called *copyleft users* and with

a commercial license to another user segment titled *customers*. Note that the arrow from the copyleft users to customers indicates that when the copyleft users extend the usage of the copylefted software they tend to reach the limits of free usage. For example copyleft concerns, commercial support and warranty requirements may attract copyleft users to buy a commercial license. To simplify the picture, there is no kind of feedback mechanism (fixes, development ideas etc.) described from software users to core product developers; those are assumed to come from the above.

Above the core product we have two developer segments. On the left is the open source *development community* which may give bug fixes and code contributions with copyright back to the core product developers (Fink, 2002, p. 40-41). On the right are commercial *development partners* who develop essential components of the core product; they may either transfer or license the copyright of the component to core developers.

3 Case Studies

Now that we have seen what dual licensing is in principle, we study it in practice with three cases each featuring a company using the model. This section starts by explaining the empirical method used and then reports the data for each company in turn.

3.1 Study Framework

In order to understand more profoundly dual licensing, three open source companies using the model were selected for detailed analysis. Company selection was based on the facts that only a few companies were known to successfully operate with a dual licensing strategy and they

also quite openly report their actions on the Internet. As can be seen from table 2, all of the companies have very popular products.

The study was conducted by collecting information from the company websites, referring to company executive interviews available on the Internet and by asking complementary questions emailing company executives directly.[2] Emailed questions addressed primarily the issues of user and customer bases, relative importance of different income sources and potential piracy problems. The following presentation is based on this qualitative data.

The presentation of each company is divided into three parts: historical background, licensing model and model effectiveness:

- The first question is how the companies ended up in a dual licensing model. At what stage of their lifecycle have they chosen dual licensing? It turns out that none of the companies have started from a dual licensing model. The concept has evolved as time has passed and both the Internet and open source software user base has grown.
- The second question is how the dual licensing model works in each case. What licenses do the companies use and for what reasons? Here we learn that while the companies may use different free software licenses they all contain a strong copyleft clause limiting the commercial usage.
- Finally, we consider the effectiveness of the dual licensing model in each case. How do the companies benefit from the free use of their software compared to the traditional software publishing model? Have the companies detected a piracy

[2] The direct questions were answered by SleepyCat CEO Michael Olsen, MySQL CEO Mårten Mickos and TrollTech VP Tonje Sund. All the questions and answers may be inquired in detail emailing the paper author.

problem? How do they manage the legal rights in a distributed and open development environment in the products they own?

	Sleepycat Software Inc	MySQL AB	TrollTech AS
Product	Embedded database	SQL database	GUI tools
Free license(s)	Sleepycat License	GPL	GPL, QPL
Users	Millions	Approx. 4 million users	Hundreds of thousands
Customers	Thousands	Around 0.1 % of users	Thousands
Main income	Licenses (>75%), services, support	Licenses (>50%), brand, services	Licenses
Development	Inhouse	Inhouse	Inhouse
Marketing	Direct and indirect	Direct	Direct and indirect
Technology	Standardized (database)	Standardized (SQL)	Non-standard (GUI)

Table 2. Several attributes of the studied open source products.

3.2 Sleepycat Software Inc.

Background. Sleepycat Software Inc. develops and markets BerkeleyDB (BDB). The product is an embedded database system that runs on multiple platforms. The first version of BDB was written by Keith Bostic and Margo Seltzer in 1991. It was released under BSD license as part of a BSD Unix distribution from the University of California at Berkeley. BDB was distributed freely on the Internet and eventually many open source as well as proprietary projects started using the product. BSD license terms allowed a wide adoption of the product even in commercial projects with no license fees to copyright owners.

As the product gained more commercial interest and more users, the programmers decided to found the company Sleepycat Software Inc. as the owner of the copyright and develop the product further. The next version added technical functionality and was therefore commercially even more valuable. It was released under Sleepycat License in 1997. From then on, BDB has been licensed under a dual licensing model. (Zimran, 2001)

Licensing model. Sleepycat's website states on their licensing policy:

> "The open source license for Berkeley DB permits you to use the software at no charge under the condition that if you use Berkeley DB in an application you redistribute, the complete source code for your application must be available and freely redistributable under reasonable conditions. If you do not want to release the source code for your application, you may purchase a [proprietary] license from Sleepycat Software." (Sleepycat, 2003)

Essentially, their licensing model is based on the usage limitations of Sleepycat License. Many companies using an embedded database in their products are simply unable to satisfy a strong copyleft clause requiring all source code of any product containing BDB to be freely available. Sleepycat's CEO Michael Olsen has described the usage of the proprietary license in an interview:

> "If a company wants to redistribute Berkeley DB as a part of a proprietary product, they can come to Sleepycat and pay us a fee to purchase different license terms from us. In that case, we sign a pretty conventional license agreement permitting use and redistribution in binary form, without forcing them to ship source." (Zimran, 2001)

Model effectiveness. The development of BDB is directed within the company. All outside contributions are implemented by company developers into the core product. Obviously, the development of a complex database engine requires understanding of the functionality of the whole. Development of add-on features is difficult. Therefore, user feedback benefits mainly

in identifying bugs and proposing new features. It would be possible to change Sleepycat License into GNU GPL but at the moment there seems to be no immediate reason for this as the Sleepycat License has been widely accepted in the free software community.

Most of the income of Sleepycat (around 75%) comes from license sales and the rest from services. Sleepycat does not promote other than license sales on their website, which is their main direct marketing channel. The usage under free license is not monitored. If license breaches are found, which is not very common, the users either buy a proprietary license or stop using the product. (Zimran, 2001)

3.3 MySQL AB

Background. The product of MySQL AB is a relational database management system. It was first targeted at web server use but is now offered also as a general database management system and specifically to users of embedded databases. The development started in 1995 by Michael Widenius and David Axmark and the first major release on the Internet followed in 1996.

At the start, MySQL shipped with its own license terms. (MySQL 1995) That license allowed limited free distribution and usage of the product with a strong copyleft term on Unix-based systems (including Linux). On Windows, the license model was shareware restricting the free use and distribution of the product. Their business model was essentially dual licensing on Unix-based systems and proprietary on Windows.

As the Linux-based version became very popular on the Internet, the free license was changed in 2000 into GNU GPL on all platforms. After that, their licensing model has been solely dual licensing. The license change limited the scope of proprietary licensing for different uses but at the same time it attracted even more users for the product. As late as in 2001, the company MySQL AB was founded to own the copyright to the database software with its partners. (Greant, 2002)

Licensing model. The product's copyright is licensed either by GNU GPL or a proprietary license. Any third party product that includes the GPLed version must be licensed under GPL. Again, commercial users may by unable to satisfy this requirement and need to opt for the commercial license. MySQL's website states their licensing model in the following:

> "Our software is 100% GPL, and if yours too is 100% GPL (or OSI compliant), then you never have to pay us for the licences. In all other instances, you are better served by our commercial licence." (MySQL, 2003)

Model effectiveness. Development is directed inside the company. As with Sleepycat, the product is very complex and can hardly be developed by third parties. In 2001, another company tried a fork but failed without being able to control the software development (MySQL, 2001). All contributions are checked and rewritten by company developers thus not diluting the copyright ownership of the product. MySQL currently includes one major component developed and licensed by a third party (InnoDB, 2003). The company estimates that they have fewer problems with free riders than proprietary software companies; the only case that has ended up in a court was with the fork.

As of today, MySQL AB receives more income from proprietary license sales than from their other income sources, branding and services. Their main income seems to come from embedded commercial users. (Codewalkers, 2002) To contrast, use on web sites – the products initial market – seems to work after the license change to GPL merely as a marketing tool for commercially licensed use on embedded products.

3.4 TrollTech AS

Background. TrollTech AS's main product is called Qt, which consists essentially of graphical user interface programming libraries. Qt can be used to develop multi-platform graphical applications. As a result, developed products embed functionality from Qt libraries.

Development of Qt started in 1992 and the company was founded in 1994 by Haavard Nord and Eirik Eng. In 1996 Qt was released under its own quite restrictive open source license, which did not allow free distribution of modifications and hence retained full development control with TrollTech. However, due to available source code Qt was selected to be used in KDE, which quickly became a very popular free desktop environment for Unix and Linux systems.

As its popularity and importance grew with KDE, pressure from the free software community to allow redistributable modification increased. In 1998 the license was changed to QPL, which is TrollTech's own inheriting copyleft license. It is somewhat similar as an idea to Sleepycat's company specific license but with more restrictions: QPL allows distribution of modifications only as separate patches. In 2000 Qt was finally released also under standard GNU GPL allowing modifications of the entire software to be distributed for free. The GPL

release was also delayed by the company founders' initial skepticism towards open source. (Fremy, 2001)

Licensing model. The licensing model of Qt is essentially the same as with the two products described above. Qt is licensed under GPL, QPL and a proprietary license. Products made with the GPLed (or QPLed) version must use the same free license. TrollTech's website states:

> "Any software produced with Qt under the GNU GPL license, and any derivatives of this software, must also be released under the GPL. As before, any user who wishes to create proprietary or closed source software must first purchase a development license from Trolltech." (TrollTech, 2003)

Model effectiveness. Development is coordinated within the company. Before the introduction of QPL, TrollTech's license terms granted that the company had full control of the development. However, now Qt also contains some code that is not owned by TrollTech AS but is rather licensed under a very permissive license from third parties. (TrollTech, 2003 b) From licensing perspective, the introduction of GPL includes the possibility of forks but in practice it seemed to result instead in an excellent marketing move. (Fremy, 2001)

TrollTech sales are based on proprietary licenses. Qt is marketed through a combination of direct sales, resellers, and strategic partners. The role of the free version is mainly to grow the user base and market the product on the KDE environment. (Fremy, 2001)

4 Legal and Economic Requirements

We have now seen how dual licensing works in principle and in practice. Next, we consider its legal and economic implications. We start this section asking how an organization using a dual licensing model should manage the rights to its product. Then, we move on to discuss the economic issues an organization using dual licensing should understand.

4.1 Strong Copyleft License and Single Rights Ownership

In every case example, the open source / free software license included *a strong (or inheriting) copyleft clause*. In essence, copyleft means that the distribution terms of the source code must be maintained even if someone modifies the source code. Inheritance means that even adaptations and derivative works must keep the license terms intact. In other words, if the source code is initially distributed free of charge then no one can charge for the source code later in any adaptation. GNU GPL and Sleepycat Licenses are both accepted as copyleft by the free software community (GNU, 2003). While QPL does not fill the strict definition of copyleft, it also essentially functions in the same way.

Also Lerner and Tirole make the same functional distinction between standard and strong copyleft. Using terms restrictive (standard) and highly restrictive (strong) licenses they present some empirical data, which shows quite interestingly that products targeted at developers tend to have less restrictive licenses. (Lerner – Tirole, 2002) The model presented in this paper contradicts the finding of Lerner and Tirole: in a dual licensing model, the software company uses a highly restrictive license in a product specifically geared towards developers for embedded use. A possible explanation for this contradiction is that the data set studied by Lerner and Tirole consisted mainly of non-commercial projects.

Another fundamental legal requirement for dual licensing is that the software company *has undisputed rights to the software product* it wishes to dual license. Ownership of rights is central because it allows company to price its software, change its licensing policy and distribute software with different licenses. A major legal risk in using open source licenses is that the license may dilute the ownership and even eliminate the possibility to dual license. Therefore, rights ownership must be managed carefully. (Välimäki and Oksanen, 2002)

The one who has written new or rewritten old software is granted exclusively copyright to the work. However, with multiple authors the copyright ownership may also become distributed. Under a copyleft license, a fully open and distributed development process without sufficient rights clearing is not suitable for any company that wishes to make any direct license sales with dual licensing. No hidden liabilities in code contributions from unknown third parties should remain.

It is possible to think of two ways to clear rights: *either by rewriting the software or by obtaining rights with a license term or specific contract.* (Välimäki and Oksanen, 2002) The first option may mean costly work but it is legally speaking the safest option. The latter option leaves the possibility of legal risks if the transfer is somehow incomplete for example because the code contributor has no authorization to withhold necessary rights. As the case examples showed, most code in successful dual licensing companies was rewritten with the minor exceptions in Qt. There, a sufficiently permissive license does not restrict the main developer (TrollTech) from dual licensing it.

4.2 Benefit and Control of Free Use

Dual licensing depends on several distinctive economic implications that must be at hand. First, there must be sufficiently large user base for the product. Here, the copyleft license enables strong *network effects* typical to information products: the value of the product to single user depends on the number of user it has (Shapiro and Varian, 1999). With the free availability and efficient distribution of the product through the Internet there are not many limitations for exponential user base growth. Especially software that depends on separate distributed components interoperating directly has strong network effect (Messerschmit and Szyperski). Shy and Thisse have demonstrated that when network effects are strong, unrestricted copying and distribution of the software product results in an equilibrium in a simplified setting within non-cooperative competitors. (Shy and Thisse, 1999)

Second, the effectiveness of dual licensing depends on *price discrimination*. A software company that manages all rights to the product may license it according to market demand. (Shapiro and Varian, 1999) For example, our case studies have shown that if there is demand for both stand-alone and embedded products, then dual licensing may be an economically viable model. Also worth noting is that in dual licensing the licensing policy – not the product features – are tailored. Only those users that have direct benefit from the use of the software are required to pay a license fee and for other users payment is more or less optional.

Third, there seems to be *no major requirements for the enforcement of copyright*. The little data we have indicates that high-end corporate users required to purchase a proprietary license also do so. Also empirical evidence shows that illegal copying of software in general has decreased during 1995-2000 relative to the software market (Osario, 2002). Open source license option may strengthen this trend. While traditional copyright licensing model is still

plagued by a vast number of unauthorized users, the free license conversely supports free usage by the majority of users who would not pay the license fee anyway. By definition, there is *no piracy* if the copying and distribution of the product is allowed. Worth to note is also that for someone who embeds a free product into a commercial one, the license purchase does not lead to the ethical and philosophical issues one may have with traditional copyright enforcement and zero tolerance. Instead, dual licensing is one answer to the *economic question* of how the copyright owner should protect his work (Rahnasto 2003, Watt 2000)

5 Conclusions

This paper has offered one answer to the question of how one can build a successful open source business model. Dual licensing – licensing technically identical product with both proprietary and open source licenses – seems to be a viable model for specific types of software companies. However, dual licensing is no silver bullet. We have identified several organizational, legal and economic *limitations and requirements*, which may in practice fundamentally limit the usability of the model.

The case studies have shown that the company as an organization needs to *believe* in the strong copyleft licensing model. At first, it may sound counterintuitive, but if the product has for example both stand-alone and embedded users then strong copyleft may be workable. Moreover, *GNU GPL license seems to be a viable marketing tool* in dual licensing. TrollTech was not convinced that GNU GPL license would allow them to continue the sales of proprietary licenses until they tried. In side, they received increasing media attention and political acceptance. MySQL had almost similar experience.

The limitations of the case studies deserve also attention. Our case examples were few in number and discussed successful companies that target their products at developers who may embed the product into a larger environment. During the company selection process *no particularly successful stand-alone end-user applications* with dual licensing model were encountered.

Finally, the paper discussed the identified legal and economic requirements for dual licensing. Legally, dual licensing requires the use of a license with *a strong copyleft clause* and that *all legal rights* to use and distribute the software are *managed by a single entity.* Economic discussion discussed how an open source license may due to *network effects* act as an efficient user base leverage. Then, the high-end of the user base may be *price discriminated* with the commercial license because of their different preferences. Finally, dual licensing may significantly *reduce the costs of piracy* and copyright enforcement.

References

Codewalkers (2002), Interview of Michael Widenius, http://codewalkers.com/interviews/Monty_Widenius.html (visited 1.4.2003)

Feller, Joseph – Fitzgerald, Brian (2002), Understanding Open Source Software Development, Addison-Wesley

Fink, Martin (2002), The Business and Economics of Linux and Open Source, Prentice Hall.

Fremy, Philippe (2001), "Interview: Trolltech's President Eirik Eng", http://dot.kde.org/1001294012/ (visited 1.4.2003)

GNU (1991), "GNU General Public License", http://www.gnu.org/licenses/gpl.html (visited 1.4.2003)

GNU (2003), "Various Licenses and Comments about Them", http://www.gnu.org/licenses/license-list.html (visited 1.4.2003)

Greant, Zak (2002), The Future of MySQL, presentation at SD Expo 24.4.2002, San Jose, CA, USA, http://www.mysql.com/information/presentations/index.html (Visited 1.4.2003)

InnoDB (2003), "MySQL/InnoDB technical support and commercial licensing", http://www.innodb.com/suplicense.html (Visited 1.4.2003)

Lerner, Josh – Tirole, Jean (2002), "The Scope of Open Source Licensing", MIT Working paper, available at http://opensource.mit.edu/ (Visited 1.4.2003)

Messerschmit, David G. – Szyperski, Clemens, Industrial and Economic Properties of Software. Technology, Processes, and Value. University of California at Berkeley Computer Science Division Technical Report UCB//CSD-00-1130, Jan. 18, 2001, and Microsoft Corporation Technical Report MSR-TR-2001-11, Jan. 18, 2001

Microsoft (2003), “Software Licensing and Development Models”, http://www.microsoft.com/resources/sharedsource/Articles/default.mspx, March 2003 (Visited 1.4.2003)

MySQL (1995), “MySQL Free Public License (Version 4, March 5, 1995)”, http://www.mysql.com/support/arrangements/mypl.html (visited 1.4.2003)

MySQL (2001), “FAQ on MySQL vs. NuSphere Dispute”, 13.7.2001 http://www.mysql.com/news/article-75.html (visited 1.4.2003)

MySQL (2003), “MySQL Licensing Policy”, http://www.mysql.com/support/arrangements.html (visited 1.4.2003)

Osario, Carlos A. (2002), “A Contribution to the Understanding of Illegal Copying of Software”, MIT Working paper, available at http://opensource.mit.edu/ (visited 1.4.2003)

Rajala, Risto – Rossi, Matti – Tuunainen, Virpi Kristiina – Korri, Santeri (2001), Software Business Models. A Framework for Analysing Software Industry. Technology Review 108/2001. Finnish National Technology Agency, http://www.tekes.fi.

Rahnasto, Ilkka (2003), Intellectual Property Rights, External Effects, and Antitrust Law: Leveraging IPRs in the Communications Industry, Oxford University Press.

Raymond, Eric (2001), The Cathedral & The Bazaar, Revised edition, O’Reilly, http://www.catb.org/~esr/writings/ (Visited 1.4.2003)

Shapiro, Carl – Varian, Hal (1999), Information Rules, Harvard Business School Press.

Shy, Oz – Thisse, Jacques (1999), "A Strategic Approach to Software Protection", Journal of Economics and Management Strategy. vol. 8, issue 2, p. 163-90.

Sleepycat (2003), "Sleepycat Software Product Licensing", http://www.sleepycat.com/licensing.html (visited 1.4.2003)

Stallman, Richard, M. (2002): "Why "Free Software" is better than "Open Source"", http://www.gnu.org/philosophy/free-software-for-freedom.html (visited 1.4.2003)

TrollTech (2003), "Qt Free Edition", http://www.trolltech.com/developer/licensing/ (visited 1.4.2003)

TrollTech (2003 b), "Licenses for Code Used in Qt" http://doc.trolltech.com/3.1/licenses.html (visited 1.4.2003)

Välimäki, Mikko – Oksanen, Ville (2002), "Evaluation of Open Source Licensing Models for a Company Developing Mass Market Software", The Proceedings of International Conference on Law and Technology (LawTech 2002), Cambridge, MA, USA, available at http://www.valimaki.org/ (visited 1.4.2003)

Watt, Richard (2000), Copyright and Economic Theory, Edward Elgar.

Zimran, Ahmed (2001), “Interview with Sleepycat President and CEO Michael Olson”, http://www.winterspeak.com/columns/102901.html, 29.10.2001 (visited 1.4.2003)