HOW SOFTWARE COPYRIGHT AND PATENTS LAWS ARE HURTING REAL INNOVATION

Alexandru ŞTEFAN

Bloomfield College, Bloomfield, New Jersey, USA

Abstract: *This paper will focus on software copyright and patents and how current copyright and patent laws are just not cutting it when it come real innovation. We will also discuss in detail the benefits of open standards any why more software vendors are moving towards it.*

Keywords: *software, copyright, patent, open standards.*

1. INTRODUCTION

This paper will focus on software copyright and patents and how current copyright and patent laws are just not cutting it when it come real innovation. We will also discuss in detail the benefits of open standards any why more software vendors are moving towards it. A formal definition of copyright is "the exclusive legal right, given to an originator or an assignee to print, publish, perform, film, or record literary, artistic, or musical material, and to authorize others to do the same".

So why is this harmful to innovation, especially in the software industry? Software is an ever-changing complex and perishable item; most applications are almost useless after 5 years. Almost all software vendors no longer provide any kind of support for software older then 2-3 years; it just isn't cost effective for them to do so. Customers therefore can't rely on using software that no longer supported, but yet paid for. This software is now considered abandonware since the company is no longer selling it. It is considered illegal now to install, edit or copy abandonware even if the hardware needed to use the software no longer exists. Developers would need to wait 75 years after the author's death before they can change a peace of software that once ran on DOS to run under vista. This is just one example of how copyright law is harmful to the secondary works that could be created by abandoned software.

Another major topic we will focus on is the topic of open standards. First let's define open standard. "An open standard is a specification that enables users to freely choose and switch between suppliers, creating a free and open competition between suppliers. To accomplish this, an open standard must have the following properties".

1. Availability: Open Standards are available for all to read and implement.
2. Maximize End-User Choice: Open Standards create a fair, competitive market for implementations of the standard. They do not lock the customer in to a particular vendor or group.
3. No Royalty: Open Standards are irrevocably free for all to implement, with no royalty or fee.
4. No Discrimination: Open Standards and the organizations that administer them do not favor one implementer over another for any reason other than the technical standards compliance of a vendor's implementation.
5. Extension or Subset: Implementations of Open Standards may be extended, or offered in subset form.
6. Protection from Predatory Practices: Open Standards may employ license terms that

protect against subversion of the standard by embrace-and-extend tactics. The licenses attached to the standard may require the publication of reference information for extensions, and a license for all others to create, distribute, and sell software that is compatible with the extensions. An Open Standard may not otherwise prohibit extensions.

7. One World: The same standard should be applicable for the same capability, world-wide. It must not be devised as a "barrier to entry" by those from other regions.

8. On-going Support: The standard is supportable until user interest ceases rather than when implementer interest declines.

9. No or nominal cost for specification. (This may soon become a requirement for no-cost specifications that can be copied further. Fees innately discriminate against many users and implementers, particularly in the third world; with the rise of the Internet fees have become a completely unnecessary discrimination).

Since copyright law is very clear in stating that secondary works cannot be created without the explicit consent of the copyright holder. Many open source applications cannot interact with copyrighted applications and a format, playing a DVD on a Linux box is a great example of this. Prior to 2001 it was illegal to play a DVD on a Linux OS because the DVD format was not licensed to any Linux application. We will have to see what happens with blue ray disks; currently there is no legal way to watch a blue ray movie on a Linux box.

A patent by default is an implied monopoly so your competitors can not bring the same product to market at a cheaper price which hurts consumers. The main reason we have patents is to spur innovation. It hurts the open standard because it reduces competition but again it's a necessary evil in order to encourage innovation. There are more arguments can be presented against patentability than for it.

First of all, patenting software inventions takes investment away from research and development. The cost of obtaining patents and defending against competitors' patents requires that significant funds be diverted away from research and development. Most software patents cover either trivial inventions or inventions that would have been obvious to persons of ordinary skill in the art at the time the invention was made. If you investigate, patents examiners rarely have a comprehensive knowledge of the specific technologies disclosed in the patent applications they examine. Developers may be forced to pay license fees for standards that are covered by patents. Let us now look at the companies that don't produce software. For them, software patents allow investment companies to purchase patents from others and generate lawsuits to collect revenue off the monopoly granted by the patent. Therefore, some believe it to be offensive that a company that doesn't create software might benefit from a patent for software. And there are many others who understand that these patents are generally purchased by highly speculative investors from software producing companies that were looking for investments.

Similar to patents, the argument for personal copyright is to grant developers temporary monopolies over their works to encourage further development by giving the developer a source of income. There are many people argue that copyrights originated only in the last few centuries. However, creativity flourished well before copyright existed. With the same thought, developers believe that ideas and knowledge should not be owned or controlled, but rather should be distributed freely throughout society for anyone's use-as long as the creator of the original idea is well acknowledged. In addition, copyrights reduce the incentive for developers to continue working, since they can receive and income by collecting license fees or royalties for popular older works instead of develop new ones. Most importantly, limiting innovation is also seen as unjust.

Many proprietary software companies have criticized the open-source software movement. These companies suggest that it undercuts, and in some cases destroys, the economic incentives necessary for the software industry to continue to create quality products by making them compete with free software. Proprietary backers and critics alike, also

argue that open-source is not sustainable because it does not provide the economic incentive necessary for individuals or teams to devote resources, such as time and effort. At the core of the pro-intellectual debate is that making software available without a price tag attached will eventually drive out proprietary software developers. As a cause, developers will not invest in creating proprietary software because they will not be able to compete with similar products available at no cost.

In an ideal world, open standards for software would be the norm. However, many software developers have been slow to adopt these standards. The arguments against open standards include protecting of research and development investment, justification of the upfront expense, providing incentives to keeping technology up-to-date, and reducing the efficiencies that may be gained by using proprietary technology to bind together complex systems.

While producers of proprietary software provide valid points, they are mostly driven by profit margins. Open standards reduce barriers to entry, facilitating a multitude of competition among suppliers. Many customers can appreciate the not being locked into a proprietary technology.

The proprietary software developers argue for more than non-standardization, they fight for profitability. The companies battle open standard are the same companies that stand to benefit the most from keeping the market the way it is (i.e. Microsoft). In actuality, standards tend to invoke innovation and create foundations for useful, compatible technologies. The greater the openness, the greater the participation - Sun Microsystems has proven this, time and time again.

It's unlikely anyone would debate the fact that incentives are beneficial for the promotion of creativity - but if the incentives are too strong, giving the initial creator too much control or control for too long, there would be little opportunity for secondary works. Essentially, it creates an under production of innovation; remove incentive all together and an under production of initial creators may occur. Law should be used to promote innovation, not the lining of a particular group's pocket. In a report written by the FTC on the patent system, they state "Patent policy is for the benefit of the public, not patent holders. The ultimate point of granting a patent is not to reward inventors, but rather to create incentives for actions - invention, disclosure and commercial development - that will further the public interest and thus benefit consumers over time".

The introduction of copyright laws was thought necessary in order to encourage creativity. It was thought without an incentive justification others could easily copy and redistributed an author's work, quickly driving the value of the work down. With an author unable to recoup their original cost or potentially profit from their work, they may have been deterred from creative expression.

Copyright laws allow creators to raise prices above marginal cost of producing additional copies of the original work. While providing an incentive to the author but creates restrictions on who can access or own the work. The excluded audience is the individuals that may have purchased the work at a slightly higher marginal cost but not at a higher price. An example of this is Microsoft's software products. Often, their products are priced incredibly high, making it difficult for some to invest in, yet necessary to own in order to be compatible with other users.

One of the most obvious examples of what unrestricted standards can lead to is the Internet. Communication across the web (the world) may not have been so wide spread or have grown so quickly without the universal access to the TCP/IP protocol. In addition HTML, the code used to power billions of web sites is an open standard. "Roughly 70 percent of the servers that seek out Web pages use open-source Apache software. Open-source Send-mail is used in 80 percent of e-mail servers." The PERL programming language is considered to be one of the most used languages on the Internet - PERL is also open standard. Other open source tools that have made enormous impacts include database systems like MySQL, browsers such as Mozilla Firefox, and the Linux operating system. While proprietary software is beneficial and has a place in the software

market, there is no denying that open source programs also play a significant role in creativity and communal development.

2. OPEN SOURCE BUSINESS MODEL PROPOSAL

When the Internet began to evolve as a medium for business there was much hype that traditional economics no longer made sense. Business models were developed that focused entirely on growth, and did not consider cash flow. These times have now changed and things are certainly beginning to normalize. What is clear however is there are some unique economic characteristics with regard to the Internet and software products that are not as persistent with traditional, tangible products. This material covers some of these issues. Business Model is the business' value proposition, and how it determines to satisfy that value proposition. Software companies need to have a specific business model. Many software companies are operating on a ROI model, presuming the web site, software presentation, serves multiple marketing functions. Alternatively search engines operate with an advertising revenue business model (PPC). Software vendors need to determine whether their business model should focus on the sale of the software, or it’s after sales support. Software marketplaces, with respect on open source software are much more efficient once a standard for the marketplace is established. Standards are able to increase the overall size of the market as the market itself increases its utility for each consumer.

Standards typically evolve in different ways:

- Proprietary Standard, owned by a single or group of companies
- Open Standard, developed by a single company (or group) and opened to the market for all to benefit
- Open Standard, developed by a consortium/industry group, oversees the future development of the standard.
- In order to help the company’s management to take a decision, with regarding the copyright/patent issues, in this chapter we will cover the differences in Intellectual Property (IP) laws and also a comparison between open source licenses based on four categories: reciprocity reach, sublicensing options, patent grant, and patent retaliation. (Table1). The categories were selected based on software management decision, with regarding the approach to open source implementation or open source software production legal coverage.

Table 1 Proprietary Model vs. Open Source Model

Proprietary Model	Open Source Model
Licensor distributes object code only; source code is kept a trade secret	Licensor distributes source code
Modification are prohibited	Modification are permitted
All upgrades, suport and development are done by licensor	Licensee may do its own development and support on hire any third party to do it
Fee are for the software, maintenance and upgrades	Fees, if any, are for integration, packaging, support and consulting
Sublicensing is prohibited, or is a very limited right	Sublicensing is permitted; license may have do distribute the source code to program and modification

A. Open vs. Closed: An Economic Perspective Conclusion

Open Source software is available for free; commercial versions of the same open source software may also available at a price. These versions include customers service, packaging, detailed instructions and free upgrades. (Red Hat's version of Linux for example.) The question is: does this pricing structure make economic sense? Should software be sold on a per unit basis to recover development costs, or sold on the basis for charging for ongoing support.

The factory, industrial age, model would suggest charging on a per unit basis for the intellectual property of the code (closed source). While this makes clear sense for

automobiles and houses, that include significant variable costs per unit, software, and other digital products tend to have very small variable costs (zero marginal costs). The costs associated with these products are fixed and sunk (development costs). Thus costs associated with sales of additional units typically are those for product support after the sale. This support is important for the product to be effective for the users in the medium - to long-term. Thus by charging on a per unit basis, to try to recover sunk costs, creates an incentive for developing software that is purchased but not used (no need for customer support). While the customer support center is considered a cost center, the after sales support will be limiting, which in turn will lead to under-served customers. A product that is given away for free, but has a paid alternative that supports the customer service infrastructure (a free alternative is required for the Open Source license for those offering commercial versions) makes perfect economic sense. This is the strategy adopted by Red Hat and this actually extends the market for the software beyond its traditional base of hackers. This argument therefore supports the economic issues related to pricing open source software, but it can also be applied to ALL types of software.

B. Why is Open Source Software important?

The general analyze conclusion comes with the following Q&A:

Q: Why is Open Source Software so important?

A: The Open Source Software Business Model

- Can be a major source of innovation
- Innovation can happen anywhere - any time;
- Development through "open communities" leads to potentially;
- Broad ideas and creativity.
- Community Approach
- Internet has changed how enterprises address technical innovation.
- Good approach to developing emerging standards
- Popular Open Source projects can become de facto / open standards;
- Wide distribution/deployment/
- Enterprise customers are asking for it
- Increase choice and flexibility - ion / use of Open Source can reduce time to market

C. Life Cycle of Open Source Process Conclusions

It is important to consider the product life cycle, when considering when to "open source" a project. Clearly some context has to be established, such that external developers are going to be interested enough to contribute their finite resources to the project. Thus an alpha version of the product needs to be complete. This was the case before Linux announced the Linux Operating System to the Minix news group. On the other hand, the project does not want to be so mature; that it is no longer interesting for external developers (their ability to contribute becomes marginal). Since they were not part of the evolutionary process of earlier development, it is hard to engage them at later stages.

An argument can be made for offering a product open source, at a later stage in the life cycle, to extend the product's life while shifting internal development resources to new development efforts. This will help guarantee the life of the product for the current installed base until they switch to the new product.

D. Recommendation

1. Open Source Software (OSS) is indeed the start of a fundamental change in the software infrastructure marketplace, and is not a hype bubble that will burst.

2. Within five years, 50% of the volume of the software infrastructure market should be taken by OSS.

3. The rise of OSS, offers the possibility that non-US players will find it easier to influence the future direction of IT infrastructure technology.

4. The differences between OSS and proprietary software are not a major factor in either improving or degrading the vulnerability of a nation's IT infrastructure.

5. We recommend that the Government obtain full rights to bespoke software that it procures - this includes any customization of off-the-shelf software packages.

6. The Open Source model offers a new paradigm for funding software in

communities-of-interest (e.g. Health and Education).

3. OPEN SOURCE ROI FUNDAMENTALS

What are the fundamental issues that organizations should keep in mind when thinking about open source ROI? In other words, what are the ground rules that underpin open source ROI assessments? Here are five key realities about open source ROI.

In order to answer those questions, key realities must be understood at first hand. While the absence of license fees is obviously attractive, the enormous amount of work required to implement an open source solution could outweigh the license fee savings of the product. Thus, all aspects of the system must be taken into account to ensure a full financial evaluation and accurate assessment of OSS ROI. Moreover, it is important to understand the soft and hard costs associated with open source. Software costs are associated with the use of internal company personnel in terms of employee time spent installing, configuring, and integrating an open source product, which imposes salary costs. These costs are difficult to calculate accurately and many organizations do not have an explicit internal cost assigned for employee time which results in a significant portion of the overall project cost. Hard costs are incurred if an outside consultant is used, or if new hardware is purchased to run the open source system. These types of costs usually require explicit payments and are easy to calculate. Consequently, it's vital to fully account for all costs associated with an open source system to ensure an accurate ROI figure is calculated. In addition to these concerns, many projects have financial calculations performed for initial implementation or for the first full year of operation. This form of financial analysis ignores a fundamental IT reality: systems have extended lifespan, since organizations are reluctant to replace a working system, given the cost and disruption associated with replacement. Consequently, it is vital to use a realistic timeframe for project ROI calculations. It is particularly important to do so for open source-based systems. Since much of the cost of these systems is in early investment for internal personnel to learn and customize the open source product, a short project time horizon can make it seem that an open source system is more expensive than a commercially-licensed product. However, because commercial products carry yearly maintenance fees, using a longer time horizon can more accurately calculate true project costs for the realistic time frame of the system. Consequently, when making an open source project ROI calculation, it is vital to use a realistic project life projection, since this will enable a more accurate ROI assessment. A common practice for ROI project planning is to create a table with ROI calculations based upon three, five, and 10 year project durations. This allows project planning to proceed with a broader range of ROI information available to assist the decision process.

3.1. The Components of Open Source ROI

Most of the costs included in calculating an open source ROI can be assigned to three primary categories. License, support, and maintenance costs reflect the payment made to an outside entity in order to gain access to use of software. License fees are well understood as the up-front payment made before a vendor delivers software, while Support/Maintenance refers to the yearly payments required so that customers may gain access to patches and enhancements for software that has already been licensed. These costs are hard costs, since they always are invoiced by an outside entity, thereby ensuring access to the software product. IT/Service provider costs are those required for technical personnel and are associated with the work of installing, configuring, customizing, and operating the software. The factors affecting the size of this cost are how difficult a piece of software is to install and configure as well as how much custom programming and integration are necessary to modify the product to meet the organization's functional requirements. Depending upon whether the technical work is done by internal or external people, this cost can be either soft or hard, or, indeed, can be a

mix of soft and hard. For this reason, it is important to carefully assess the amount of work necessary to fully implement an open source system. Organizational costs are those borne by the end users of a new system. Often referred to as “cognitive load,” these costs reflect the learning curve imposed by new software. When a user begins to utilize new software, productivity goes down, since he or she is unfamiliar with the new system. The overall cost experienced due to this temporary reduced productivity is captured in Organizational costs.

3.2. Open Source ROI Scenarios

A. Custom system vs. OSS

When an organization internally develops custom software system it produces software that meets its exact requirements, rather than using an externally developed package that forces it to compromise on its requirements. However, the downside of custom development is also clear: it’s expensive to write custom systems from scratch. Open source in this case is a wining strategy because it is distributed with binaries and source code rather than binary format alone, which is typical of commercial software products, thus enabling companies to use it as components within its internal system and add its organization-specific code to the core product. In this fashion, it can still achieve its aim of creating a system that meets its exact requirements while avoiding the cost of developing the entire system top to bottom. IT/Services costs are much lower when using open source software in this case because rather than absorbing all development costs itself; the company is able to take advantage of the investment by other developers and organizations in the open source product. In addition to the cost savings associated with development, there are ongoing savings as well made possible by the use of community of developers that fix bugs and enhance the software at no charge for the company. By contrast, should a company develop a custom system internally; it will take on the entire expense of bug fixing and product enhancement. Organizational costs may also be lower when using open source software since many other organizations have offered feedback about the software to the open source developers; this feedback has enabled them to improve the usability of the product. By contrast, a custom-built piece of software will have to begin receiving and incorporating feedback upon initial release, which means that the software will not be as user-friendly as an open source counterpart. It is therefore recommended that organizations should prefer use of open source software in this scenario to reach higher ROI, but must be careful in assessing whether the requirements of the company mandate internal development.

B. New system vs. OSS

If an organization is considering implementing a system that utilizes technology the organization has no experience in using, then it will need to invest in training and skill development whether it uses a commercial or an open source product. For this reason, the IT/Services costs are similar for both open source and commercial alternatives. Similarly, because the system is new, the Organizational costs experienced upon system introduction are equal. After all, whether the system uses commercial or open source software, users will be learning a new application; therefore the Organizational costs will reflect the learning curve costs experienced by the user base.

The primary ROI difference in this scenario is the cost of the software itself. While the organization may choose to engage a commercial open source vendor to provide a subscription or support for the selected open source product, it is still likely to realize significant savings in comparison to the license fees associated with the commercial counterpart to the open source product. Typical savings available from open source subscription versus commercial license range from 75% to 90%, indicating the potential savings by selecting open source. The general strategy for this scenario is to consider open source as the default choice, and examine the detailed financial situation in order to make a final decision.

C. Commercial vs. OSS (Dissimilar technologies)

Sometimes an organization can consider replacing an existing commercial software system with an open source alternative that is based on a different technology, such as replacement of a Windows server with a Linux machine. While it is clear that this scenario has the potential to save License/Maintenance fees, the savings may not be as dramatic as they would appear at first glance. Since the initial license fee for the commercial software was paid in the past, it is a sunk cost that is unrecoverable at the current point in time. In this case, the ROI would depend on the cost of ongoing maintenance fees compared to the subscription or support fees possibly associated with the open source alternative available to the organization. However, the IT/Services and Organizational costs for the open source alternative may be significantly higher if an open source choice is made because the organization has already made a significant investment in skill development and overcoming the user base cognitive load associated with the existing system. Rather than absorbing such a significant cost all at once due to a cutover to the new software, organizations often adopt a "Surround and Extend" strategy, in which the existing commercial software products are left in place and new open source products are implemented to provide complementary functionality. In this way, the organization can begin to realize the benefits of open source, while gradually investing in building IT/Services skills as well as user skills. Consequently, this scenario often poses ROI challenges in making an open source choice. A common example of this approach is the use of Linux as a file/print server in a Microsoft Windows Server infrastructure. The existing Windows Servers continue to provide application functionality, while the company begins to use Linux in a low-investment, low-risk fashion. The duration of the project is also critical in this scenario, since the initial costs of building IT/Services skills, implementing the system, and educating the user base must be compared against the total amount of commercial software maintenance fees to be paid during the lifetime of the project. Different project duration assumptions may dramatically change the ROI of the project, so a complete analysis of the project is important for this scenario.

D. Commercial vs. OSS (Similar technologies)

The circumstances are significantly different when the two alternatives share a common technology basis. No additional investment in skill development is necessary in this scenario. The organization has already made that investment for the existing commercial product, which means that little additional investment is necessary when shifting to a new open source product. This is a common experience when comparing standards-based products, because the two products are bound to resemble one another, given that they both implement the same standard. For example, if an organization already has a significant installed base of a commercial Java application server like Web Sphere or WebLogic, the cost of moving to a comparable open source product like JBoss is likely to be quite small, since most of the skills necessary for the commercial products will transfer quite easily to the open source product. In this scenario, the IT/Services and Organizational costs are the same for both the commercial and open source alternatives; the primary difference between the alternatives is the cost of ongoing commercial maintenance compared to any subscription costs for the open source product. The recommended strategy is to seek out these opportunities very aggressively, since they present very high ROI potential.

REFERENCES

1. www.linuxmagazine.com;
2. www.infoworld.com;
3. Cope, R., and Stormy, *Integrating OSS into Your Environment*, February 2006; www.linuxworld.com;
4. David, M., Wednesday, W., *Open source software and the future of the world*, 06 February 2008; www.itwire.com;
5. www.rfgonline.com;
6. http://www.dwheeler.com/essays/opendocument-open.html.