Open Source Software Strategies for Venture-Funded Startups

David Wood
MIND Laboratory, University of Maryland
College Park, MD 20740, USA
Technical Report No. TR-MS1287
15 February 2005

ABSTRACT

Should your startup release all or part of your product under an Open Source license? Should you use Open Source Software in your commercial product or contribute to Open Source projects? How can an Open Source Software strategy help your startup company with cost reductions, marketing, public awareness or code stability? How does Open Source Software relate to your patent strategy? In this paper, an experienced entrepreneur suggests Open Source Software strategies that can help capitalist companies make money while contributing to the community.

Keywords: Open Source Strategies, Business Strategies, Software Licenses, Entrepreneurship

1. INTRODUCTION

A primary goal for cash-strapped startups is often the cost reductions inherent in using Open Source Software. Paying to reinvent the wheel or licensing of expensive proprietary libraries are typically not options for such companies. But "free" software does have a price to businesses: one must comply with license restrictions, be cautious when combining code with different licenses and be clear as to the business value created. Contributions to Open Source projects may also make clear sense, especially to avoid the cost of developing similar frameworks from scratch, but will generally need to be addressed as part of an overall intellectual property strategy.

It is no longer uncommon, though not routine, for startups to consider releasing some or all of their core software under an Open Source license. The benefits to such a strategy include opportunities for greater public awareness of a company's existence, the ability to perform stealth marketing at conferences, a reduction in the overhead associated with evaluations and the potential for bugs to be repaired (and even features developed) by third parties. All of these advantages must be consciously compared to the loss of secretiveness and to what degree a business benefit is obtained from holding proprietary knowledge closely.

A common misconception regarding Open Source Software is that a releasing company forfeits ownership and copyright of its source code. Practical or actual forfeiture may or may not occur, depending upon the Open Source license used. This paper will suggest which Open Source licenses to avoid and which are particularly appropriate for startups. The relationship between intellectual property strategies (inclusive of software patents) and Open Source licensing will also be addressed.

Strategies to segment a commercial product into Open Source and proprietary components will be addressed. An effective segmentation can serve several purposes: Academic researchers and commercial users without budget for software purchases may use an Open Source project without impacting limited startup sales teams or requiring support assistance. Meanwhile, proprietary extensions geared to a particular market can assure that sales will not be lost to one's own Open Source project. A real-world case study will be used to illustrate this strategy.

2. USING OPEN SOURCE SOFTWARE

Startup companies have limited budgets and limited time to show value. Venture-funded startups are under particular time pressure from their investors. Use of high-quality Open Source components avoids hard costs since Open Source projects do not require royalty or license payments.

What about quality? It is certainly possible to find poorly-written Open Source projects. However, many Open Source projects have been shown to be of high quality (e.g. [1]), making them particularly well suited as foundational components. This is especially true for those projects which provide common infrastructure, such as scripting languages, Web servers and development tools. The author has watched several companies reinvent tools and components at their own cost where better quality Open Source projects existed.

For these reasons, engineering teams should be encouraged to build proprietary software using carefully-chosen Open Source components. Startups should be particularly careful to put their available engineering effort where it will provide the most value to their business.

Using Open Source components will ensure that third-party bugs will not halt development of your core product. Bugs may (in all likelihood will) be encountered, but may be found and repaired in the available source code. That is good, but therein lies a dangerous situation. As soon as a bug is fixed in your copy of an Open Source project, you have forked, or split the code base. The only way to ensure that the fixed bug stays fixed in the next version of that code is to submit it back to the Open Source project. Failure to do so will benefit no one. The time it takes to submit a fixed bug should be considered against the downstream benefits of maintaining your own code.

Open Source code comes with licenses. Users, including commercial companies, must comply with those licenses. The bad news is that not all software licenses are compatible.

Therefore, it is best to track all licensed software used and perform periodic reviews of licenses for compatibility. Engineering managers should ensure that any new component has its license reviewed prior to investing significant integration work.

3. RELEASING OPEN SOURCE SOFTWARE

Startup companies may benefit from the release of technology under an Open Source license under certain circumstances. Some of these circumstances are:

1) When a company needs to "make a market" for a technology (Open Source projects often have wider audiences than commercial products, especially when they implement "bleeding edge" ideas);
2) When a company wishes to test the market for the viability of a technology (It is easier to get people to try something that is free);
3) When a company requires a large set of testers (such as for a security product);
4) When a company is working with a standards body which requires an Open Source implementation;
5) When a product is based on rapidly evolving academic research (Academics will often contribute to Open Source implementations);
6) When a company is facing significant threats from Open Source projects (Sponsoring an Open Source project allows a company to control features and determine market needs);
7) When a company wishes to cheaply expand market awareness for a related ("pro version") product.
8) When a company faces large numbers of evaluators and relatively few buyers (Servicing evaluations can be expensive and time consuming).

Companies deciding to create an Open Source project should remember that it is an expensive activity to properly support. Failing to properly support an Open Source project will only damage the sponsoring company's reputation and will not serve commercial goals. An Open Source project should be treated as a major part of a company's routine, marketing program and budget.

Users of Open Source Software will generally not expect commercial-quality installers, documentation, training or support (but they will be more than happy to absorb them if offered). All of those may be offered commercially as value-added features of a commercial product. However, users will expect to see a reasonable Web site for an Open Source project, including updated news of new features and releases, a knowledge base including access to bug reports and mailing lists for both users and developers. It will be expected that engineers will spend company time participating on these mailing lists.

The more professionally an Open Source project is operated, the more its users appreciate it. Word of mouth from Open Source users is often better than any other form of advertising for a startup company. If an Open Source project develops a solid reputation among users, a commercial "pro" version will tend to drag that reputation. The reverse also holds.

Open Source users often include engineers and systems administrators at potential customers. The opinion of these people will be key to any future sales opportunities since they will be asked to evaluate products. Companies in many industries, including financial services, manufacturing and system integrators, have advanced technology groups (ATGs) which are often responsible for evaluations. Members of those groups will generally be familiar with Open Source offerings.

Can an Open Source project be pulled back after it is created? One cannot legally rescind Open Source licenses already issued, but it is a simple matter to take down a Web site which may be the only download source for a project. Of course, others may choose to mirror a download site and are free to do so under any Open Source license. Whether that happens is up to chance.

Sourceforge [2] offers free online hosting and management software for Open Source projects. Using Sourceforge is often the fastest and least expensive way to host Open Source projects. It is worth noting that Sourceforge [2] will not allow administrators of an Open Source project to delete any project or its source code once a single download has taken place. That may be advantageous if you wish to protect a project's existence.

It is important to any company's sales and marketing teams that an Open Source project not compete with the company's core commercial products. One way to manage this is to bring the sales and marketing teams into discussions of specific features to be released under an Open Source project. Sales and marketing staff and product managers, who spend their days speaking to potential customers, will generally have a very solid understanding of which features customers are willing to pay for.

Giving away software can also change relationships with customers. The Business Software Alliance claims that the retail value of pirated software exceeds $13B annually [3]. The only way to pirate Open Source Software is to fail to comply with its license. Customers are actively encouraged to make copies of Open Source Software in order to encourage its wide adoption. Open Source users become "up sell" targets for startups; this is especially important given startups' traditionally low sales volumes. Selling support, services or add-ons to those customers are also options.

Open Source projects often rely on significant internal salesmanship. It can be an uphill battle to convince board members and senior management that any technology should be given away instead of sold. Only a convincing business case based on the economics of a particular company can convince those people to provide consistent support for Open Source projects.

4. PRODUCT SEGMENTATION

If a company decides to release part of a product under an Open Source license, the question naturally arises where and how to draw the line between Open Source and proprietary features. Sales, marketing and product management staff will tend to want to minimize Open Source features and engineers will (or should) demand that any product segmentation be readily implementable.

Splitting a code base can be a costly endeavor. Engineering managers should be asked for an estimate of the costs involved prior to implementation. A code split will cost both time and money.

The cleanest and simplest way to manage two offerings is to ensure that the Open Source project is a legitimate subset of the commercial product. That is important for two reasons: (a) It allows the commercial product to be built on top of any working build of the Open Source project and (b) It is easy to implement changes or additions via well-defined interfaces. That approach makes a code split relatively easy for an engineering team to manage and therefore allows the engineering team to be more responsive to feature additions driven by sales, marketing and product management staff.

5. OPEN SOURCE SOFTWARE LICENSES

There are many forms of licenses for Open Source Software. The Open Source Initiative (OSI) [4], a non-profit corporation which tracks and recommends such licenses, lists 58 Open Source licenses on its Web site. It should therefore not be a surprise that few entrepreneurs, engineering managers or software engineers are expert in the legal subtleties. This has lead to significant confusion in the marketplace. Where and when is it safe to incorporate Open Source Software into a commercial product?

It is always best to thoroughly research any software licenses which you choose to use and acquire a documented legal opinion regarding them. This paper provides the briefest of overviews of Open Source licenses.

Defining Open Source
The OSI provides a general guideline for all Open Source Software in their Open Source Definition [5]. There are ten requirements for a software license to meet the OSI criteria. Besides accessible source code, such software much be able to be redistributed and must be allowed to be modified. The software must, as stated in various ways, play nicely with all parties to the contract.

Note that Public Domain Software ("Freeware") is different from Open Source Software. Software released into the public domain has no license. Anyone may do anything they want with it. All software which falls under the term "Open Source" has a license to which one must comply.

Those violating the terms of an Open Source license open their organizations to threatened or actual lawsuits. Companies releasing Open Source Software are not giving software into the public domain; they retain rights in accordance with the license and may choose to defend those rights against violators.

Types of Open Source licenses
In spite of the large number of Open Source Software licenses, they fall generally into one of four categories. This is because many of the licenses are simple variations on a theme. The categories are:

1) BSD-Style. Named for BSD Unix, licenses of this style only require copyright notices to stay attached to the code. BSD-Style licenses are thus very close to public domain (e.g. [6]). Code licensed this way is often used within commercial products.
2) GNU-Style. Named for the Free Software Foundation's GNU Project, GNU-Style licenses require code to "stay free", by making users legally liable to provide source code to it. Also, the GNU General Public License [7] (used for Linux and many programs which are distributed with Linux) contains significant restrictions when code is combined with other code; these restrictions generally prohibit using GNU GPL code within commercial products.
3) Mozilla-Style. Pioneered by Netscape Communications, Mozilla-Style licenses were designed to provide commercial protections for companies wishing to release an Open Source product. Intellectual property protections are explicitly defined. Mozilla-Style code may be used within commercial products, although the source code must be provided.

Best licenses for companies using Open Source Software
Commercial companies wishing to use Open Source Software in their commercial products should thoroughly review their licenses. Generally speaking, BSD-Style and Mozilla-Style licensed code may be used within commercial products as long as one complies with the license restrictions. Some GNU-Style licenses (namely the GNU Lesser General Public License, LGPL) may also be used in commercial products.

Code licensed under the GNU General Public License (GPL) may not be combined with commercial code unless particular technical care is taken to separate the code components. This separation must cause the two programs to run separately (that is, they may not be "linked"), although they may communicate (e.g. via a socket).

Best licenses for companies releasing Open Source Software
Commercial companies wishing to release code under Open Source licenses should proceed carefully. Releasing code under some Open Source licenses (such as BSD-Style licenses) is tantamount to forfeiting all rights to it, other than attribution and continued use. Releasing under GNU-Style licenses is very popular in some circles (due to its use by the Linux operating system and most associated applications), but those licenses fail to address certain issues of importance for startups. In particular, startups wishing to pursue a patent protection strategy must ensure that their Open Source license does not invalidate their patent applications.

Two particular licenses are recommended to provide startups adequate intellectual property and market protections. Both are Mozilla-Style licenses: The Mozilla Public License (MPL) version 1.1 [8] and the Open Software License (OSL) version 2.1 [9].

The MPL specifically provides for patent licenses to be issued to Open Source users, thereby protecting software patents in a way BSD-Style and GNU-Style licenses do not. This is particularly important to venture-funded startups who will be required by their investors to have a defined intellectual property strategy. Similarly, the MPL addresses copyright and trademark issues of the releasing company. Lastly, the MPL requires modifications to source code which are distributed to third parties to be documented and submitted back to the copyright holder. That reduces the ability of commercial competitors to leapfrog the originator with a competing product.

The OSL is very similar to the MPL but leaves out the requirement for modifications to be submitted to the originator. For this reason, most startups will prefer the MPL. However, the average life span of a startup is short. The MPL's provisions for an "Initial Developer" (the releasing company) work best when the releasing company is stable and is likely to continue to be present in the market. Startups anticipating dissolution should seriously consider the GPL or Lesser GPL (LGPL). The latter allows licensed code to be included in commercial software products.

Version numbers of these licenses are important and the newer versions should generally be used. The modifications made in the newer versions are generally the result of further legal review. An exception may be the GPL version 3, which may take some time before becoming widely accepted.

Lawyers and Open Source Software licenses

Sadly, most intellectual property lawyers are ignorant of Open Source licenses. Startups who turn to such lawyers for advice may find themselves charged for the many hours of reading necessary for a lawyer to comprehend all 58 OSI-approved licenses! This is clearly to be avoided.

Startups should ask their law firms whether they can provide expertise in Open Source licenses, or whether they will acquire it on their own time, prior to engaging them for a license review. Alternately, another law firm with Open Source expertise may be used. A third alternative is to clearly outline the goals a company is trying to accomplish by using a particular Open Source license and limit discussion to whether those goals are likely to be met.

It is important to ensure lawyers know that they do not have the option to make modifications to the standing Open Source licenses. Even minor modifications would disallow the use of the OSI "Open Source" service mark and raise questions in the Open Source community, both of which would be contrary to the company's interests. Lawyers should be warned that the each OSI license needs to be accepted or rejected as a whole to avoid paying for modifications which may not be used.

Only in rare cases should a new Open Source license be proposed and it should then be presented to OSI for approval. In most cases, it is wise to take advantage of existing OSI-approved licenses, which have already undergone substantial legal review.

6. PRACTICAL CONSIDERATIONS

According to an unpublished study by a US venture capital firm [10], only one venture-funded startup per 27 in the US mid-Atlantic region successfully raised a second ("Series B") venture investment in 2004. The author has heard the ratio 1 in 30 being used for the New York venture market in 2005. Most startups simply fail. What happens to the ownership of intellectual property after a corporate failure?

Most startups cease operations, many without a formal bankruptcy. If the startup has a working product which has shown market value, an asset sale is possible. Even if an asset sale raises cash, that money is likely to pay off creditors and venture (preferred) share holders. It is unlikely in the extreme that founders or staff will see benefit. What, then, is the incentive for entrepreneurs to tightly hold proprietary intellectual property? The release of some or all core technology under an Open Source license can ensure future access to the code following cessation of operations.

Venture capitalists may also benefit from an Open Source strategy. VCs tend to invest in complimentary technologies in the hope that their portfolio companies can help each other succeed. Portfolio companies may still benefit from Open Source technology after a company closes.

As shown in the following case study, release of Open Source Software does not preclude an asset sale.

7. CASE STUDY

The author was recently co-founder and CTO of a venture-funded startup. The company was headquartered in the United States with a five-person sales, marketing and management team and a twelve-person engineering team. The company created a proprietary product for the enterprise middleware market. The product implemented standards of the World Wide Web Consortium (W3C).

The company had the following goals:

a) To "make a market" around new W3C standards.
b) To track, and preferably influence, W3C standards as they evolved.
c) To protect its intellectual property rights, including its patent applications.
d) To leverage the significant amount of ongoing research occurring in academia.
e) To not directly compete against no-cost solutions; Several Open Source software projects were discussing the development of competing technologies.
f) To foster inexpensive but effective market awareness.

The company decided to pursue a composite Open Source/proprietary licensing strategy. The product was therefore split between an Open Source component and a proprietary commercial product built on top of the Open Source project. Composite licensing addressed the issues in this way:

(a) was addressed by having two offerings aimed at two different markets. The Open Source project was aimed at academics, corporate researchers and the W3C standards process. The product added features required by commercial and government production users (security, distributed queries, additional APIs, enterprise management functionality, documentation/examples, training, support, etc).

(b) and (c) were addressed by the company's involvement in W3C standardization efforts, the existence of the Open Source project and the careful choice of the Open Source project's license. The Open Source project allowed the company to participate in the W3C without fear of the W3C Patent Policy [11]. The company carefully choose the Mozilla Public License version 1.1 [8] in order to provide patent and competition protections.

(d) was assisted by the existence of the Open Source project. Open Source contributions in the form of ideas, algorithm discussions, research directions and bug fixes were critical to

the productization effort. The company's small team was greatly assisted by these contributions. An Open Source project helped to keep the product on the leading edge.

The company was particularly careful where the line was drawn between proprietary commercial and Open Source software. Features were individually selected for appropriate licensing prior to each release by the product manager and CTO and approved by the lead salesman and the CEO. The proprietary portion of the source code addressed critical needs of production, not research, users.

(e) was a significant problem for the company. As many as five Open Source software projects were actively discussing the development of similar efforts. The release of the Open Source project allowed the company to directly impact the competing efforts. It also allowed the company to choose which features were released as Open Source. This was a subtle but incredibly powerful form of market influence. The careful choice of the Open Source project's license ensured that competitors (commercial or Open Source) had to provide the company with any changes to the Open Source project. Competitors could learn from the Open Source project, but not easily leapfrog the company's proprietary efforts.

(f) was addressed by releasing a substantial Open Source project and participation in the W3C standardization process, allowing the company to become a thought leader. This lead to market opportunities which would have otherwise not have been accessible. Market awareness was fostered by linking the company's Web site from the Open Source project site (but not vice versa). Presentations to academic conferences regarding the Open Source project were able to mention the company's name and support for the Open Source project.

The company decided not to offer commercial support for the Open Source project. That decision helped to foster market demand for commercial sales.

The commercial product and the copyright to the Open Source project were successfully sold in 2005 to one of the company's customers, a large defense integrator. The purchaser has been impressed with business development leads generated via the Open Source project and has decided to continue supporting the project.

8. CONCLUSIONS

Strategies were presented to ensure the safe and effective use of Open Source Software by startup companies. Particular attention was paid to the requirement to comply with Open Source licenses.

The major types of Open Sources licenses were discussed and two (the MPL and OSL) were highlighted as most appropriate for venture-funded startups. Two others (the GPL and LGPL) were suggested for companies facing dissolution. Strategies for managing legal counsel were provided.

Practical considerations for segmenting commercial product offerings with Open Source components were given. Finally, a case study was presented which illustrated how an actual venture-funded startup managed its intellectual property, inclusive of Open Source components.

Taken together, these strategies provide guidance to entrepreneurs, board members and business and engineering managers of startups for the effective use of Open Source Software.

9. REFERENCES

[1] The Open Source Initiative, http://opensource.org/
[2] The Open Source Initiative, Open Source Definition, http://opensource.org/docs/definition.php
[3] Regents of the University of California, The BSD license (amended), 1999, http://opensource.org/licenses/bsd-license.php
[4] Free Software Foundation, Inc., The GNU General Public License, 1991, http://opensource.org/licenses/gpl-license.php
[5] The Mozilla Foundation, Mozilla Public License version 1.1, http://opensource.org/licenses/mozilla1.1.php
[6] Rosen, Lawrence E., Open Software License version 2.1, http://opensource.org/licenses/osl-2.1.php
[7] González-Barahona, Jesús M & Gregorio Robles, Free Software Engineering: A field to explore, **Upgrade: The European Journal for the Informatics Professional**, Vol IV, No. 4, August 2003, http://www.upgrade-cepis.org/issues/2003/4/up4-4Gonzalez.pdf
[8] Core Capital, http://www.core-capital.com/index.htm
[9] World Wide Web Consortium, W3C Patent Policy, 2004, http://www.w3.org/2004/pp/
[10] Sourceforge, http://sourceforge.net/
[11] Business Software Alliance, http://www.bsa.org/