To open source or not to open source

Adam L. Beberg
President, Mithral Communications & Design, Inc.
April 2000

Open source represents both opportunities and threats that need to be taken into account when planning the future of any software project. Adam L. Beberg examines the times when open source is a good idea, and when it's not, from a business perspective.

The open source movement has grown to a level of religious devotion, and this may have scared you away from using open source in your projects. But the open source movement has gained momentum for many reasons: it's idealistic, coders work on fun things, and it gets everyone low or no-cost software complete with source code. Keep in mind that no matter what the zealots say, there is no one way to define the terms and no right answers.

The fact is, open source isn't an all-or-nothing thing. There are times when open source is the only sensible choice; there are other times when open source doesn't make much sense at all; and there are more ways to apply an open source license to your code than you might have realized.

The bottom line

Assuming that you are a company considering open source for your own code, then the bottom line is revenue. If the software can't make money, you can't pay the bills, and everyone will need a new day job. The choice to open source has many implications to revenue, so the implications need to be considered as primary reasons for or against open source.

If you do open source, revenue purely from the software is all but eliminated. This leaves only publishing (sticking the software in a pretty box), tech support when the user gets stuck, and consulting for people that need help setting up your software as sources of revenue. Unless your product is many megabytes, hard to use, and hard to set up, then these may not be enough to cover development and marketing costs.

Open source does give you some advantages that offset the inability to sell the software: bug hunting, reliability, and community. Since the source is out there, if there is a bug, anyone can hunt it down and then send the fix back in to be incorporated into the master copy. With all of this bug-stomping, your software will end up being very stable and reliable; this is offset by all the semi-functional "neat stuff" that users will want to add.

While the open source community can be a large source of programming talent, do not open source your software because you expect them to solve your problems for you. You will still have to do all of the design, initial development, maintenance, and product documentation. The desire for programmers to fix any bug that doesn't directly affect them or to add a feature that they don't personally want is very low. Any sufficiently hard bug to track down will result in a bug report, not a patch. User-submitted "features" will tend to be neat things that they and a few other people may use, but all core functionality will still need to be developed by your programming team.

Benefits of open source

There are many areas where open source is a simple and wise business decision from any angle.

Legacy applications that are no longer a part of current products being sold are an excellent area for open source, assuming that no third-party software is incorporated into the product. Old applications that are no longer cost effective to maintain or support can be made open source as a way to abandon the product, while not abandoning the users (as users who feel abandoned typically become ex-customers in a very short time). This way, the users who cannot migrate to newer products get the source and can use it to fix or modify things to suit their needs, while your entire development staff can move on to current projects.

The specifications to hardware and device driver code is another obvious place where publishing more information will benefit the product. In the past many vendors withheld the specification on how to work with their hardware fearing it would give competitors an advantage. The only real effect this had was preventing many users from having the option of buying their hardware. Writing device driver code is a development cost that is only made up by selling more of the hardware. Publish everything you can about it, and even help to get drivers written for the various open source operating systems. This will help you sell more hardware to anyone who wants it.

Any product that is aimed at establishing a standard protocol or API needs to be open source. If the source is available, then the adoption of any protocol or API will be much faster, and the programmers who use it will be far more comfortable with it. Many modern protocols and attempts at standards that were not open source failed before anyone even noticed they existed. In these cases the software itself does not represent the value; the things it enables does. Often, the amount of revenue from consulting, training, and licensing from the additional developers will completely dwarf any revenue that would have been gained by keeping the code proprietary. Many large Fortune 500 companies make 100 times more revenue on their consulting and support contracts than they do on their hardware and software sales. The key to this case is that the protocol, API, or hardware is worthless until a developer takes it and uses it to create value for the user.

Open source can be used as a powerful attack and defense mechanism while competing with rivals. If a rival's application has no need for support or consulting to use it, and an open source version of that application can be created and released, then the revenue from the closed application will be removed. This is very similar to the old method of discounting a product until a competitor is driven out of business, or salting the earth in a war. On the other hand, if you can develop your application as open source, then a rival will have no way to attack you in a similar way.

The realms of closed source
There are still many areas of the software world where open source has not made much progress. In these cases, open source projects are rare, and those that do exist are not a threat to commercial developers.

Game and entertainment software is the first area where cost is not an object, and open source alternatives rarely exist. People want the latest and greatest entertainment software, and are happy to support the developers if it will get them more frames per second, more players in the game, and more realistic blood splattering and gore. The developers of this software are also so in demand that anyone showing signs of free time and talent would be hired instantly. If the user is not willing to pay, then they will either pirate the entertainment content or simply buy one of thousands of other entertainment alternatives. The driving factors in this category are that the applications are more entertaining to use than they are to write, and it requires constant work to keep up with the state of the art.

The realm of scientific software, large complex applications, and applications that require a Ph.D. or years of experience to even contemplate working on are also not of general concern to the open source world. Open source projects succeed when large numbers of coders each contribute something to a project. Since the supply of extremely skilled people is very limited, and they are often already working 60+ hour weeks, they do not generally have any time left to contribute to open source projects unless the project has gone commercial. As a general rule, if the average to advanced coder can develop an application, there are probably several open source versions of it already. If the average coder cannot begin to work on the software, then it is likely there are only commercial versions of the software.

The other area that is not very open source friendly is boring software that is not sexy to develop, or applications that need end-user support. Things like word processors, spreadsheets, and other end-user tools. It's no secret that most open source software is not documented in a way that an end-user can use. You don't tend to see applications where the user is likely to need to call in for help for something that is being given away for free. Until open source projects started forming companies and paying people to develop all the boring applications non-programmers actually needed, there was no sign of them.

One thing to remember no matter how you license your source is that it takes about 1/20th the time to copy and reverse engineer software then it takes to develop it the first time. Your competitors, be they another company or an open source project, will be on your heels from the start unless you have something they can't do. Usually, things that they won't be able to do include hardware devices, patented technology, or a reliable service and support infrastructure. This has been true since the dawn of time, but in the past you could at least turn around and compete on price. Those days are gone. Now, once they catch up, your revenue is in danger. But they won't be making any money either, so if the stock market bubble funds your competitors but not you, then you may have to worry - unless you use open source or other tactics to head them off.

Innovation issues
One thing that open source is not generally known for is innovation. In general, the rate of innovation in the closed source world is far faster because the desire to stay ahead and stay in business has always been stronger than the desire to re-implement something that's already been done. Even the operating system gems of the open source world are just starting to gain the security and critical enterprise reliability features common in the high-end commercial systems 10 years ago.

All coders by nature like to do new and interesting things, but not necessarily things that are really new, just things which are new to them personally. A simple search for any common application will result in dozens of nearly identical open source projects to build it. Like an army, there are a few high-skill generals and masses of low-skill soldiers working their way up the ladder by working on projects like the ones that the generals worked on when they started out. Projects have high overlap, with low innovation.

If you have an application you need to build and there is an existing open source project to build it or something like it, then join that project or hire some of the main people in the project to work on it full time. The application can then move to the level where service, support, and consulting become sources of revenue. There is a long history of projects that started out as research or open source and turned into commercial ventures.

Licensing issues
If you do decide to open source your software, you will face the decision of choosing a license or writing your own. If one of the existing blessed open source licenses (see Resources) is acceptable, it will save you having to go through the approval process and months in a very uncomfortable spotlight. Put on your flame-proof suit, because you will encounter fanatics no matter what license you choose to use, but it is your software and you get to decide how to license it.

Unlike commercial licenses, open source licenses have never really been tested in court. So far peer pressure and threats of bad press have been enough to enforce them. No one is sure what will happen when the licenses are finally tested in the courts, not to mention the courts of countries with laws different than the United States. Odds are no license (open source or otherwise) means anything due to the patchwork of international laws applied to a global Internet.

Open source meets the real world
There is a rapidly growing middle ground in the licensing of software to deal with the interests of companies and authors while

also having the spirit of open source.

Hybrid licenses like the Apple Public Source License, or the Sun Community Source License (see Resources) are not considered "pure" open source by the older factions of the open source community. The licenses still protect the ownership, patent, and other interests of the inventors but publish the source and make it usable to many people. The open source community initially objected to these new licenses, but eventually licenses evolved that were reasonably acceptable to both commercial inventors and the open source community.

Dual licensing is also a common option. One way is to release everything under both a traditional commercial use and an open source license. Users who wish to use the code commercially can provide revenue while providing the benefits of open source to noncommercial and academic users. The other way dual licensing is done is to lead with the current version under a commercial license, and follow later releasing old, obsolete versions under an open source license. Again many paying users will need the most current version; others can wait for the version to be made open source. Dual licensing seems to be more acceptable to the open source community since it has been done for a longer time.

Final tips
Avoid using the terms "open source" or "free software" completely. They seem to tap too many raw nerves and cause confusion. Simply refer to the license by name.

Use an existing license if you can, but even this will bring criticism. Expect it, and don't let it disturb you.

Not everyone speaks English; and lawyer-babble doesn't translate. Keep it simple and in common language no matter what license you use.

Keep the marketing team at bay so that they don't aggravate the open source community by saying something "wrong."

Summary

In the end, open source is one software licensing option, which like any other, has advantages and disadvantages. The path is tricky for commercial interests to walk down. Experience shows that neither pure commercial or pure open source licensing works in most modern cases. More and more things are being done in that fuzzy middle of the real world. Open source companies are acting more like traditional companies, and software under dual or hybrid licenses is growing.

Resources

- GNU Public License (GPL)
- Mozilla Public License
- Apple Public Source License
- Sun Community Source License
- The temple of the "open source" definition and the list of "blessed" licenses
- The developerWorks open source zone

About the author

Adam L. Beberg is the President of Mithral Communications & Design, Inc. and is best known for his work in the field of distributed computing and with Cosm. In 1997 Beberg founded distributed.net which solved several encryption contests. In April 1999, he left distributed.net to return to the Cosm project's goals to create a general purpose distributed computing infrastructure. Beberg holds a degree in Computer Engineering from the Illinois Institute of Technology. He can be reached at beberg@mithral.com.

What do you think of this article?

○ Killer! ○ Good stuff ○ So-so; not bad ○ Needs work ○ Lame!

Comments?

Submit feedback