



[Tech Notes are short articles discussing library-related technology.]

[Tech Note:] The Five Biggest Myths About Open Source Software

Before discussing the myths it would probably be useful to define open source software (OSS). Open source software is software for which the source code is available. Source code is the human-readable form of computer programs. Among the languages encoded in source code are C, C++, Perl, Python and many others.

Now for the 5 most prevalent myths about OSS:

1. *Open source software is free.*

It is true that most OSS is free of acquisition costs. Some OSS *does* cost money upfront. For example, WiLS uses an accounting package called Traverse. When you order the product you receive the source code, but you also receive a bill for thousands of dollars.

What good is it to have the source code for Traverse? Not much, in this case, because there is no culture of independent programmers writing revisions to the code. It's pretty much the company that created Traverse that does the modifications, as well as some consulting companies. However, if the originating company were to go bankrupt or cease supporting Traverse the source code would offer customers the opportunity to hire programming talent to maintain the orphaned software.

The case of Traverse is the exception rather than the rule. As I've mentioned, most OSS is free of acquisition costs. However, any package with large-scale functionality, such as the open source integrated library systems (ILSes) Koha and Evergreen, faces a number of costs before they moving from the free download to functioning on a user's site. The software needs to be compiled and configured before it's ready for prime time. This may range from extraordinarily simple to extraordinarily complicated. Often, the more flexible a package, the more tweaking of it is required. ILSes are no exception.

Who does the massaging and tweaking of an open source package of software? A current library staffer, e.g., a systems librarian, can often do the setup or configuration. In the case of Koha and Evergreen there are companies that provide expertise for a fee. Even if a library is able to perform the configuration in-house, the cost in staff time is still an expense, and that cost may be significant.

Additional costs may include backups of data, updates to the software, tracking of bugs and seeing to their resolution. These all consume staff time.

Having said all of this about the apparent and hidden costs of some OSS, I must note that there are many simpler pieces of software that are free for acquisition and aren't

particularly expensive in terms of staff time. These tend to be small packages that accomplish a single or small set of tasks. Many can be easily installed. A good example is Zotero, “a personal research assistant inside your browser.” This is an easily-installed Firefox extension. Source code doesn’t come with the standard download, but it is accessible from the Zotero site. Most people will find that Zotero is satisfactory as-is, but the site has a forum especially for feature requests. And for the very ambitious, it’s possible to modify the source code to turn it into an “Offspring of Zotero.”

So, sometimes open source is cheap, sometimes it is expensive. Your needs and capabilities dictate which it will be.

2. *All open source software is good/All open source software is bad.*

These are two faces of the same coin. Some people take up the banner and mount a campaign promoting the cause of open source. Others opine that OSS is bad software. Alas, both are right. As in many cases in life the truth lies somewhere between these polar opposites.

How can you determine which is good and which is bad? The final determinant is your experience in using the software; if it is reliable and predictable in its operation then it is good OSS. Otherwise, it is bad.

Especially for an expensive OSS application you may want assurance *before* trying it out. There is no completely foolproof method to determine if it’s right for you, but there are some indicators that point in one direction or the other. The most obvious is popularity: if a piece of software is used by many people or institutions it is presumably useful. Of course, word-of-mouth can also be a useful measuring point. Ask colleagues who use the software personally or at their institution and find out what they think. Does the software perform as promised? What are its weaknesses or quirks? Is there some means to propose changes or report bugs?

There are other pointers to software quality. If there is a website for the software, does it have active blogs or forums? If so, it’s a good indicator that feedback is welcome. Are announcements or news about the software on the web site current? Old news may suggest that, like the web site, the software itself is not being frequently updated. Of course, the most obvious indicator of frequency of updates is the software itself. If it hasn’t been updated recently that may be a warning sign.

Is the software the effort of only a single developer? It may be difficult to find out how many developers there are unless the web site or download site explicitly indicates the authors. If there is only a single developer for a complex program, this may indicate a problem. First, the software may be or become idiosyncratic, reflecting the developer’s interests and design ideas; this may be good or bad. Second, what happens if the single developer loses interest in the software or has no time to maintain it? Or if she becomes sick or for some other reason can no longer work on the package? If there is only a single developer it is possible no one else will carry on with updating the software, even with source code readily available.

OSS is neither inherently and uniformly bad nor good. Just like commercial software, some software is better than others. *Caveat emptor.*

3. Open source software is not fully-featured.

As I've alluded to above, some OSS is fully-featured, some is not. Check to see if the package has a web site; you can find out about which features are available there. You can use the methods outlined above, particularly user comments, to determine the usefulness of the software for your purposes.

As with commercial software, particularly "shareware" (software sold at a small cost), there is a great deal of software that is designed to accomplish one function or a small group of functions rather than a whole range of tasks. Many do a good job of this, some do not. With a simple, single-function package it is much easier to download the software and get it going immediately with minimal or no setup.

Again, the last word is *caveat emptor.*

4. There is no technical support for open source software.

I think I've covered this well already. The quality of support differs, usually depending on how active the developer community is around the software. If there are many people involved, your report of a bug may be handled rapidly. Of course, there are also cases where an individual author, with no other colleagues working on the source, will quickly respond to any problems.

Documentation may be fragmentary or completely lacking, and responses to questions will usually depend, once again, on how rich the developer community is for the software. Often, for complex programs with a large installed base, the best configuration documentation will come from people who are users and who have been through the process themselves.

For some packages of wide interest, consultants are available, e.g., LibLime for Koha, but of course this is not free.

5. There is a cadre of volunteer programmers behind every piece of open source software.

There *is* a cadre of volunteer programmers for some open source software, e.g., the GNU/Linux operating system and Drupal content management system. Most widely-used open source products have a core of developers who contribute to the program code. Less prominent products have fewer people working on them. As I alluded to above, there are a lot of open source software packages that are the work of one or two people. In any case, having a cadre of developers doesn't assure that your desire for a new feature will be met by instant action. Conversely, one or two developer projects may accommodate you immediately. So, even if there is a cadre of volunteer programmers it

doesn't provide complete assurance that your needs will be met immediately, although bug fixes will probably be attended to rapidly.

Takeaways From This Article

- Open source software comes with associated costs, but acquisition cost is usually not one of them.
- Open source software, like any other software, must be evaluated to insure both that it works and that it will meet your needs.
- It is important to determine how much technical support will be required for an open source package, and whether your operation can wait for support. If not, it may be appropriate to contract with a consulting company to insure uptime.
- Some (not most) open source software is “not ready for primetime.” *Caveat emptor.*

Many thanks once again to Jane Richard for her excellent editing skills. Any errors or infelicities which remain are, of course, mine.

—Tom Zillner (tzillner@wils.wisc.edu)