The Open Source rEvolution - A Pragmatic Approach to Making the Best of It

Terry White
EDS Fellow
26533 Evergreen, MS 1007
Southfield, MI 48076
001 248-262-7911
Terry.White@eds.com

Joel Goldberg
EDS Consultant Sr.
26533 Evergreen, MS 1007
Southfield, MI 48076
001 248-262-7097
Joel.Goldberg@eds.com

Lori Scharich
EDS Client Delivery Manager
5505 Corporate Drive, MS 5004
Troy, MI 48098
001 248-267-7841
Lori.Scharich@eds.com

1. Why Open Source?

When embarking on a computer systems development activity you are faced some basic alternatives to get the job done:

1. Reuse existing software
2. Reuse and modify (to meet additional requirements)
3. Build to own (or sell)
4. Buy to license
5. Buy to own
6. Some combination of the above

It used to be that building your own was your only choice. As companies built massive libraries of computer systems, and other companies found that they could sell their computer systems for a profit a whole industry was born. Most people would rather borrow or reuse something that someone else already had than build it themselves.

Sharing routine sets of computer instructions, computer programs and computer software among programmers and users has been around since the dawn of computer programming. Why would anyone want to write a software solution from scratch when someone else has probably already written it?

Arguments for reusing developed program code include:

- "its already written"
- "it works"
- "its less expensive or even free in some cases"
- "maybe they know something I don't"

Arguments for not reusing developed program code include

- "I can make a better one"
- "it doesn't do everything I need"
- "its not fast enough"
- "its not as efficient as I need"
- "I want something my competitors don't have"
- "I need a guarantee it will work".

The first set of arguments tends to come from pragmatic programmers who just want to get the job done. They understand the value of reusing existing code and would rather spend their time on creating something truly new. The second set is influenced by business realities and desires. The business is concerned about ensuring efficient operations, competitive differentiation and sustainability. Of course some of the views may overlap, especially the build a better one. But, generally the arguments against reuse are driven by a desire to be different in some way – usually in a good way.

Software development has come a long way since computers were first invented, and so has reuse. Today its possible to buy software solutions in any shape and size, from low-level functions to complete enterprise systems. Of course you can still build a software solution yourself, and also use trusted pre-built software components to help you quickly put it together. Reuse is still very strong in the software community and what once was just a handful of programmers sharing code with their friends is now supported by technology - the Internet. This technology wonder allows virtual communities of programmers to collaborate and share their work from anywhere in the world.

2. Organizational Support

Organizations like Apache, Mozilla, GNU (GNU's Not Unix), and others have emerged to allow collaboration and sharing of computer program logic and ideas among talented individuals who want to build a better piece of software and share it with anyone who wants it. These are "Open-Source" software organizations. They provide a structure that enables everyone to work together in an organized manner. Supported by the Internet, collaboration & sharing tools such as SourceForge (http://sourceforge.net/), and a governance structure that includes a generally accepted work process, these developers are able to build both simple and complex computer programs and then offer them up to anyone.

In some instances Open Source software can be developed very quickly because many people are continuously looking at it and evolving it. It really depends on the size and skill level of the participating developer group and the importance of the project to the collective developers.

Groups and individuals who decide to use their code can do so free of charge but must agree to a very simple license agreement that explains how it can be used and the fact that it comes without any warranty. Open source software isn't necessarily industry standard software, although there are some implementations, such as BIND software that allows Internet web sites to be referenced with a name rather than a number, which have become a de facto standard. Open Source software isn't necessarily an off-the-shelf product either. Open Source software is usually developed out of a general need and can take the form of a complete system, a very small component or utility, or even a set of foundation level modules that can be used to develop larger solutions.

Examples of Open Source as complete systems are the Mozilla browser, Linux Operating System and the Apache web server. Examples of lower level components or foundational modules include the Log4J logging library, and the Struts model-view-controller framework for constructing web applications with Servlets and Java Server Pages (JSP). It's estimated that about 60% of the software that drives the Internet is based on Open Source software. Many commercial software products have embedded pieces of Open Source software or used Open Source software as a starting point for a larger application.

3. "Standards"

Open Source is "Open" because it is available to anyone who wants to contribute and be a part of any of the efforts that are in progress. The Open Source software is freely distributable and can be further changed or enhanced by those who use it. Open Source is free from adhering to a direction coming from companies seeking proprietary gain. Some involved feel that it's a case of the underdog and underprivileged standing up against threats from corporate monopolies. *It's not open because it's a standard and it's not an Open Standard.*

If a piece of Open Source software becomes wildly popular it may eventually find its way to being a standard implementation. Many pieces of Open Source software are clever tools that make developing computer software easier. It is important to note that many Open Source developers do follow industry standards when developing their Open Source software, and that Open Source software is subject to a peer review process.

Still, Open Source software is without warranty, and maintenance and support is not included. The hope and goal is that "Open Source promotes software reliability and quality by supporting independent peer review and rapid evolution of source code. To be OSI certified, the software must be distributed under a license that guarantees the right to read, redistribute, modify, and use the software freely." (from OpenSource.org)

4. Decision Factors

People choose to use Open Source implementations for a variety of reasons. The least of which should be that "its Free". The real questions to ask are:

- How does the open source software compare to its commercial and proprietary counterparts?
- Is it widely adopted?
- Is the participating developer community large? (or just one person?)
- Can I find people who know about it and therefore can it be supported over time?

It should also be evaluated against traditional software selection tests in order to ensure it meets the necessary requirements. Consideration also needs to be given to the expected life cycle of the software – is this a short-lived activity or is a large investment being made that requires a payback over time?

Furthermore, when choosing Open Source software to be used as the basis for a commercial product or service offering it is important to outline a strategy for its selection, use, upgrade and support. Open Source software should be treated as if you wrote it. There is no other party that will be held responsible if it does not perform as expected or something goes wrong.

5. Strategies

Different strategies may be needed for software that forms the basis for a system, and software that is used as the system. Some Open Source software can be treated and used as a "black-box" which contains all the required functionality. Other Open Source software provides functions that contribute to a larger system. In both cases upgrades, maintenance and support must be considered.

In the case where the Open Source software provides a foundation it is likely that modifications will be made to the original Open Source software over time. These need to be carefully managed to ensure that they do not diverge from the Open Source software project charter and direction. Straying from the original charter may prevent the ability to take advantage of enhancements to the base software.

Release schedules for Open Source software is often very frequent, and can be as frequent as daily. These releases will contain new features and will likely contain bug fixes. Open Source software is still developed by people and people make mistakes. Don't assume that all of the code has been tested and tested in a manner that exercises all of the features that you will use. There is no substitute for testing your implementation of the Open Source and testing your finished product.

Finally, if you or your company decides to use Open Source software, it is a good idea to become a member of the organization or foundation you are obtaining the software from. This will put you on the mailing lists for enhancements and bug fixes, allow you to see the feature wish-lists and provide the means for you to influence the direction of the Open Source projects that you have an interest in.

6. Conclusion

Open Source seems like a Revolutionary concept. Upon closer examination it's an Evolution of good software development citizenship that has leveraged, and exploited, the available technology to expand beyond the boundaries of a person's workplace or intimate circle of friends.

It's the freedom from constraining schedules and deadlines, and traditional ROI models that permit the Open Source developers to be a little more creative than their business alter egos.

It's the expansion of software developer capacity to collaborate on, design, and build software while keeping pace with the competition created by for-profit businesses that make Open Source a viable direction.

The real threats to Open Source are the business realities of competition, risk, profit, and loss.

Businesses need to take a pragmatic approach to using Open Source software by comparing it to the competition, judging it on its capabilities, and understanding both the benefits and the risks that it brings.