Open Source Software Adoption: A Status Report

Huaiqing Wang, *City University of Hong Kong*
Chen Wang, *StockSmart*

Open source software has emerged from the hacker community, but because of many misgivings and myths regarding its maturity, making informed adoption decisions is hard. Systematically applying requirements-oriented criteria to open source software offers a practical roadmap for navigating this new landscape.

Using the right software is increasingly critical to project success,[1,2] but the choices keep getting wider and more confusing. Open source software has entered the mix, leaving the traditional confines of the hacker community and entering large-scale, well-publicized applications.[3] However, although some argue that it is ready for wide-scale commercial adaptation and deployment,[4] the myriad number of OSS packages make actual adoption a real challenge. This article presents a straightforward and practical roadmap to navigate your OSS adoption considerations.

We do not have a universally accepted definition of OSS. For instance, Netscape, Sun Microsystems, and Apple recently introduced what they call "community-source" versions of their popular software—the Mozilla project, Solaris, and MacOS X, respectively.[5,6] Such efforts, while validating the OSS concept, also make their inclusion into the OSS community a potential topic for contention. Here, we will use the loose definition of OSS that includes publicly available source code and community-source software.

Requirements-oriented considerations

Commercial IT development today is vastly different from that of 10 years ago; all-encompassing, proprietary in-house software development has effectively disappeared. Many efforts now focus on integrating off-the-shelf software packages to achieve particular software implementation goals.

You must consider many requirements when choosing a suitable software package, regardless of whether the candidate is open source or commercial.[5–7] Most of these criteria are common and have been extensively studied. We will not cover the following important adoption criteria because they do not distinguish between OSS and commercial-software candidates: functional capability, efficiency, speed of execution, and organizational standards and preferences. Specifically, our criteria apply a product-oriented evaluation framework in which we can compare and analyze distinctive features of OSS candidates.[5,7] We will emphasize the technical and managerial

0740-7459/01/$10.00 © 2001 IEEE

requirements in which the nature of OSS is particularly relevant. These two aspects correspond to the two classes of stakeholders in commercial IT efforts.

However, our aim is to outline the various requirement considerations rather than present their relative importance—you must prioritize the criteria with respect to your particular project. For instance, in the case of a legacy-system data conversion project, future upgradability might be moot but high reliability would be critical.

Technical requirements

A potential OSS would have to be evaluated according to several technical requirements involving architectural, development, and operational issues.

Availability of technical support

To adopt an OSS candidate in a commercial IT effort, you must have commercial-grade technical support available (at reasonable cost). This includes training, documentation, real-time support, bug fixes, and professional consulting as needed. To enable the development team to get off to a quick and smooth start, having a binary distribution of the OSS widely available is preferable so that the initial familiarization process can occur seamlessly.

Future functional upgradability

If the target application is to be operational, maintained, and extendable, the new software must be upgradable to provide additional capabilities. As a result, the current and future status of your OSS's development becomes a significant factor, because continuous development and bug fixing enable future upgrade capabilities. In addition, backward compatibility is important so that future versions of the OSS require minimal recoding and reintegration with existing system functionality.

Open-standard compatibility

For a large and complex IT project, all the components must adhere to a particular open standard or protocol. It is insufficient that the OSS adhere solely to the various open standards at any point in time. It must also have continuous development momentum to adhere to future revisions of the standards as they evolve.

Customizability and extensibility

For an OSS candidate to be adopted, it must be flexible enough to be customized or integrated in widely different technical environments. The package might also have to be extended to include extra, potentially proprietary functionality. While OSS is generally considered highly customizable and extensible—as the source code is publicly available—you must take into account the complexity of the effort to make such modifications at the source level. Also, you must consider the OSS package's dependency on operating systems, development tools, and other software packages that might significantly affect OSS extensibility. Whether or not you can integrate the OSS with commercial software is also an important factor, because all the software must be able to be integrated with other software packages.

High reliability

For an OSS candidate to be considered operationally robust and highly reliable, it must have been operational in a large number of applications and its performance evaluated and reviewed. For critical systems, you would be prudent to adopt software that has been widely used commercially instead of one that has yet to gain sufficient operational data and use analysis.

Management requirements

From a project management standpoint, a potential open source or commercial candidate would have to meet various resource allocation, licensing, and maintenance requirements to be adopted.

Budgetary

For the most part, OSS is considered free in the sense that generally no or minimal costs (for example, shipping and handling) are involved. However, there are indirect costs, including development, technical support, and maintenance efforts. For most IT projects, indirect costs can grow larger than the original package purchase cost.

Development team expertise

It is critical to consider the development team's existing expertise with Unix, Perl, or other OSS technologies. Lack of familiarity here would require extensive team retraining and the adoption of not only new soft-

Our criteria apply a product-oriented evaluation framework in which we can compare and analyze distinctive features of open source software candidates.

Table 1

Open Source Software Licenses and Their Effects

	Can be mixed with nonfree software	Proprietary modifications can be made private	Can be relicensed	Allows proprietary licensing
GNU Public License	Y	N	N	N
Library GPL	Y	Y	N	N
Berkeley Software Development	Y	Y	N	N
Community Public License	Y	N	N	Y
Commercial	Y	Y	N	N

ware but a new development philosophy as a whole—resulting in significant cost and resource consumption.

Licensing and project scope

Adopting OSS is not free from the terms set forth by software licenses. OSS products have several different types of license, each of which imposes a different set of restrictions that could potentially impede critical project capabilities such as internal reuse, proprietary custom extensions, and resale. Table 1 lists the following common types of OS license: GPL (GNU Public License), perhaps the most common one; LGPL (Library GPL), a modified version of GPL applying specifically to software libraries; BSD (Berkeley Software Development), applying mostly to derivatives and variants of BSD Unix; and CPL (Community Public License), a type of license typically found in community versions of commercial software. The licensing terms of your chosen software will affect your current and future project scope, such as internal use versus resale.

Long-term maintainability

Almost all operational IT projects must be maintained over time, so it is important to consider the complexity of maintaining the software you adopt. OSS characteristics such as development status, standard adherence, and the availability of support all affect the long-term manageability of your project.

Analyzing OSS characteristics

The following list describes 10 OSS characteristics and the possible values we can assign. By assigning a value to each of these characteristics for a particular OSS, we can specify that software's capability to meet its requirements. We did this with a representative collection of OSS that is either widely used or widely noted in technical periodicals, including the community-source versions of Sun's Solaris and Apple's Mac OS X. Table 2 presents the resulting chart.

1. Technical support: the amount of available support for the OSS.
   - – Support limited to direct, ad hoc individual developer support.
   - + Support based on community-oriented group support.
   - ++ Support tied to one or more commercial entities providing comprehensive support for the OSS (for example, Red Hat provides complete support for Linux, and Cygnus supports all GNU packages (interestingly, Red Hat acquired Cygnus in November 1999).
   - — No longer being developed or supported.
2. Backward compatibility: the effort required by an existing system to maintain compatibility with the OSS.
   - – OSS is either in its first stable release or its functionality has been modified such that systems using a previous version would require significant effort to upgrade to the current one.
   - + A moderate effort is required to upgrade to the current version.
   - ++ Virtually no effort is required to upgrade to the current version.
3. Standard compatibility: The open standard that the OSS adheres to and that multiple vendors have agreed to.
   - OSF (Open Software Foundation).
   - DNS (Domain Name System).
   - ANSI (American National Standards Association).
   - LDAP (Lightweight Directory Access Protocol).
   - SSL (Secure Sockets Layer).
   - SMTP (Simple Mail Transfer Protocol).
   - X11 (X-Windows Protocol).
   - HTTP (Hypertext Transfer Protocol).
   - HTML (Hypertext Markup Language).
   - SQL (Structured Query Language).
   - MIME (Multipurpose Internet Mail Extensions).
   - N/A: does not follow any open standard.

Table 2

Open Source Software Characteristics

		Technical							Managerial			
		Technical support	Backward compatibility	Standard compatibility	Binary availability	Integration with commercial SW	Commercial adoption	Open source dependency	Software license	Current development status	Commercial substitutes	Notes
Operating system	BSD	+	++	OSF	Y	+	+	n/a	BSD	Stable	Y	freebsd.org
	Linux	++	+	OSF	Y	+	++	n/a	GPL	Stable	Y	linux.org
	Macintosh OS X	++	–	OSF	Y (binary only)	–	–	n/a	CPL	Commercial release	Y	apple.com
	Solaris (announced)	++	++	OSF	Y (binary only)	++	++	n/a	CPL	Commercial release	n/a	sun.com
Application environment	Bind	+	++	DNS	Y	+	++	Unix	BSD	Stable	Y	isc.org/bind
	Gnome	+	–	n/a	Y	–	+	Linux, BSD	GPL	Stable	Y	gnome.org
	GNU CC	++	++	ANSI	Y	++	+	Open platform	GPL	Stable	Y	gnu.org
	GNU Emacs	++	+	n/a	Y	+	+	Unix	GPL	Stable	Y	gnu.org
	GNU Make	++	++	n/a	Y	+	+	Unix	GPL	Stable	Y	gnu.org
	Java	++	+	n/a	Y (binary only)	++	++	Open platform	CPL	Stable	n/a	javasoft.com
	KDE	–	–	n/a	Y	–	–	Linux, BSD	BSD	Dev. release	Y	kde.org
	Perl	++	+	n/a	Y	++	++	Open platform	BSD	Stable	N	perl.org
	Sendmail	++	++	SMTP	Y	+	++	Unix (OS version)	BSD	Stable	Y	sendmail.com
	Tk/Tcl	++	+	n/a	Y	+	–	Open platform	BSD	Stable	N	scriptics.com
	X-Windows	++ (vendor)	++	X11	Y (vendor supplied)	++	++	Unix	BSD (X)	Stable	N	x.org
Development library	Gimp	–	–	n/a	Y	–	–	Unix	GPL	Dev. release	Y	gimp.org
	JDK	++	++	n/a	Y (binary only)	++	++	Open platform	CPL	Stable	n/a	Javasoft.com
	LDAP	–	n/a	LDAP	N	+	–	Unix	BSD	Disc.	Y (Netscape)	Defunct
	OpenLDAP	–	–	LDAP	N	–	–	Open	BSD	Dev. release	Y	openldap.org
	OpenSSL	–	+	SSL	N	–	–	Open	BSD	Dev. release	Y	operssl.org
	SSLeay	—	+	SSL	N	–	–	Unix	BSD	Disc.	Y	mozilla-crypto.ssleay.org
Application	Apache	+	++	HTTP	Y	+	++	Open	BSD	Stable	Y	apache.org
	Mozilla	+	+	HTML	Y	++	–	Open platform	CPL	Stable release	Y (Netscape)	mozilla.org
	MySQL	+	+	SQL	Y	–	–	Unix	CPL (Recent: GPL)	Stable	Y	mysql.org
	PHP	+	+	n/a	Y	–	–	Open platform	BSD	Stable	Y	php.net
	Pine	+	+	SMTP, MIME	Y	–	+	Unix	BSD	Stable	Y	pine.org

4. Binary availability: official or unofficial binary distributions are available. Even when an official distribution is widely available, there might be extensive unofficial binary packages that do not receive the same level of support and release upgrades as the official source-level and binary packages.
   - Yes.
   - No.
5. Integration with commercial software: the extent to which the OSS has integrated with commercial software.
   - – Virtually no widely used commercial software can be integrated with the OSS.
   - + A moderate number of commercial software can be integrated with the OSS, but no commercial installation history exists.

For More Information

opensource.oreilly.com

Although the O'Reilly Associates Web site aims to provide an overview of available books, it is also an excellent central location for general information regarding the state of OSS.

www.redhat.com

Red Hat is an great centralized source of all OSS information related to Linux (an open source version of Unix that is rapidly gaining popularity in commercial and noncommercial environments). The site is geared to all technical backgrounds.

www.sourceforge.net

SourceForge is a free service for technology-savvy users. The source code for almost all current OSS is available here, except for well-established OSS such as Apache, PHP, Linux, and the like. Be prepared to dive directly into source code and source-related documentation.

- ++ Many commercial software integration possibilities are available and have been deployed in commercial environments.

6. Commercial adoption: the extent to which the OSS has been commercially adopted.
   - – Virtually no commercial entity has adopted the OSS.
   - + A few commercial entities have selected and installed the OSS.
   - ++ The OSS has a large installed user base.
7. OS dependency: the specific operating systems on which the OSS depends; if available for virtually all major ones, it is designated an open platform. Although no OSS operating system is compatible with any application designed for commercial operating systems, almost all the OSS environments, libraries, and applications have been ported to commercial operating systems, except for the packages still under development (KDE, Gnome, Gimp) and Unix-specific applications (Bind, Pine).
   - Unix.
   - Linux.
   - BSD.
   - Open platform: available for virtually all major operating systems, including the various flavors of Unix (Linux, BSD, Solaris, and others), Windows, and Mac OS.
8. Software license: the OSS's licensing format. The differences between the following types of licenses are the type of modifications and integrations an implementation party is permitted to perform on the OSS (see Table 1).
   - GPL (General Public License): applies to all OSS applications developed by the Gnu organization.
   - LGPL (Library GPL): covers the various libraries developed by the Gnu organization.
   - BSD: includes all derivatives of the BSD license, such as the X-Windows license "X."
   - CPL: includes various community-source projects.
9. Current development status.
   - Development release: The OSS is still being actively developed and features added.
   - Stable: A stable, widely installed version of the OSS exists, with ongoing development efforts underway.
   - Discontinued: OSS development efforts have effectively stopped.
10. Commercial substitutes: whether commercial substitutes exist for the OSS.
    - Yes.
    - No.
    - N/A: commercial vendors offer community-source versions of the software; there is a corresponding commercial-software flavor, such as the commercial Netscape Browsers and the community-source version of Mozilla.

If made a mere five years ago, Table 2 would have contained virtually no commercial adoption or commercial-grade technical support for almost any of the OSS reviewed. Over the last five years, OSS has made giant strides in improving overall stability, support, and compatibility (for more information, see the related sidebar). Nevertheless, only a minority of the representative OSS set now have commercial-grade support and commercial adoption. Continued improvement in these areas will no doubt make other OSS candidates competitive for adoption in commercial IT projects.

Open source software has become a legitimate choice for commercial adoption, as its use in Internet applications shows. As OSS continues to mature, it will play an increasing role in the software industry.

What does this mean for the OSS devel-

opment community? Besides jumping for joy, we hope that our work reflects some of the areas that require improvement for a more rapid adoption of OSS by the commercial IT entities. Announcements such as funding for Covalent Technologies,[8] a commercial venture targeted specifically at supporting the commercial users of the Apache Web server, show that the OSS community is paying increasing attention to improving support, licensing, reliability, and other areas. We believe that such efforts will ensure the continuing success and innovation of OSS in the future.

References

1. C. DiBona, S. Ockman, and M. Stone, *Open Sources: Voices from the Open Source Revolution*, O'Reilly & Associates, Cambridge, Mass., 1999.
2. E.S. Raymond, *The Cathedral & the Bazaar*, O'Reilly & Associates, Cambridge, Mass., 1999.
3. "The Netcraft Web Server Survey," Sept. 2000; www.netcraft.com/survey (current 21 Feb. 2001).
4. T. O'Reilly, "Lessons from Open Source Software Development," *Comm. ACM*, vol. 42, no. 4, Apr. 1999, pp. 33–37.
5. A. Brown and K. Wallnau, "A Framework for Evaluating Software Technology," *IEEE Software*, vol. 13, no. 5, Sept. 1996, pp. 39–49.
6. A. Schamp, "CM-Tool Evaluation and Selection," *IEEE Software*, vol. 12, no. 4, July 1995, pp. 114–118.
7. E.A.Giakoumakis and G. Xylomenos, "Evaluation and Selection Criteria for Software Requirements Specification Standards," *Software Eng. J.*, Sept. 1996, pp. 307–319.
8. "The Venture Capital Report," *Forbes*, 15 Dec. 1999, www.forbes.com/1999/12/17/mu4_print.html (current 22 Feb. 2001).

About the Authors

Huaiqing Wang is an associate professor of information systems at the City University of Hong Kong. He specializes in research and development of intelligent systems and Web-based intelligent agents and their e-business applications (such as multiagent-supported risk-monitoring systems, intelligent-agent-based knowledge management systems, modeling, and intelligent Web-based educational systems). He received his PhD in computer science from the University of Manchester. Contact him at the Dept. of Information Systems, City University of Hong Kong, Kowloon, Hong Kong; iswang@is.cityu.edu.hk.

Chen Wang is chief technology officer for StockSmart, which provides aggregated real-time financial information. He was previously a cofounder and CTO for FirstCircle. His primary industry-related research interests include cryptography, Internet commerce, open source software, privacy, and agent-based technologies. He has a BS in computer science and has completed graduate work in information systems at the University of Toronto. Contact him at StockSmart,116 John St., Suite 801, New York, NY 10005; cwang@stocksmart.com.