Evaluating Open Source Software

Matthew Kennedy

There is an overwhelming amount of open source software available today that can be used throughout the software development life cycle. Nowadays, it is generally not a matter of whether one should use OSS, but rather, where

Kennedy *is a professor of software engineering at DAU. He served in the U.S. Air Force as a network intelligence analyst and he has more than 10 years of experience in information technology. He has a bachelor's and master's degree in computer science.*

one should use it. If one were to start a new software development project, he would probably begin by looking for various types of software to aid in development, such as an integrated development environment, version control system, and a bug tracking tool, to name a few. If he looked exclusively for OSS, he could use Eclipse for the integrated development environment, Subversion for the version control system, and Bugzilla for the bug tracking tool. Those products are available for download and are open source. Looking outside the development environment, one's deployed system may require a database. A person could use a proprietary database such as Microsoft® Access, Microsoft Sql Server, Oracle®, or an open source option such as MySql. When looking to fill a technological need, OSS may be a viable option.

In July 2008, the U.S. Air Force Office of Advanced Systems and Concepts funded Georgia Tech Research Institute to create and release an open source version of FalconView. Used by the Department of Defense since the 1990s, FalconView is a comprehensive mapping tool that supports various mapping formats and includes ample map analysis tools. With both government and private applications moving to open source development, the proper evaluation of OSS throughout the program is imperative to making informed decisions that could affect the life cycle of the project. What are some of the factors that must be considered when choosing whether to use OSS?

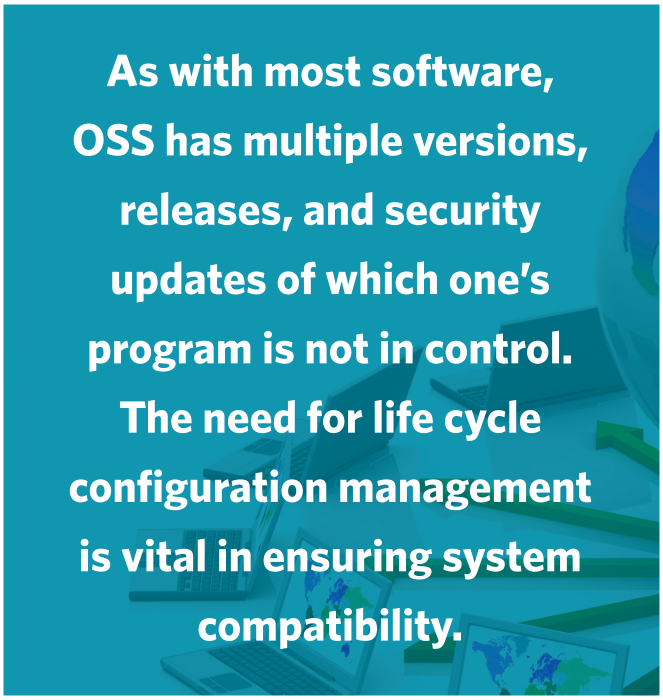
What is OSS?

According to a DoD chief information officer memorandum of 2009, "Clarifying Guidance Regarding Open Source Software (OSS)," OSS is "Software for which the human-readable source code is available for use, study, reuse, modification, enhancement, and redistribution by the users of that software."

That definition of OSS could apply to various terms used throughout federal and DoD guidance and directives. The Federal Acquisition Regulation/Defense Federal Acquisition Regulation Supplement defines commercial computer software as "Any item, other than real property, that is of a type customarily used by the general public or by non-governmental entities for purposes other than governmental purposes, and (i) Has been sold, leased, or licensed to the general public; or (ii) Has been offered for sale, lease, or license to the general public."

Chapter four of the Defense Acquisition Guidebook defines non-developmental software as "Any software that is not legacy software for the program, or is not developed as part of the effort being accomplished by the developer team. NDS includes COTS software, government furnished software, open source software, and software being reused from another program."

These definitions show that although OSS is not explicitly defined in DoD guidance and directives, the terms already in place clearly fit.



As with most software, OSS has multiple versions, releases, and security updates of which one's program is not in control. The need for life cycle configuration management is vital in ensuring system compatibility.

Some open source software projects are as big as, if not bigger than, their proprietary counterparts. According to its website, MySQL, an open source database application, has had more than 100 million copies of its software downloaded or distributed throughout its history and is currently on release 5.1.

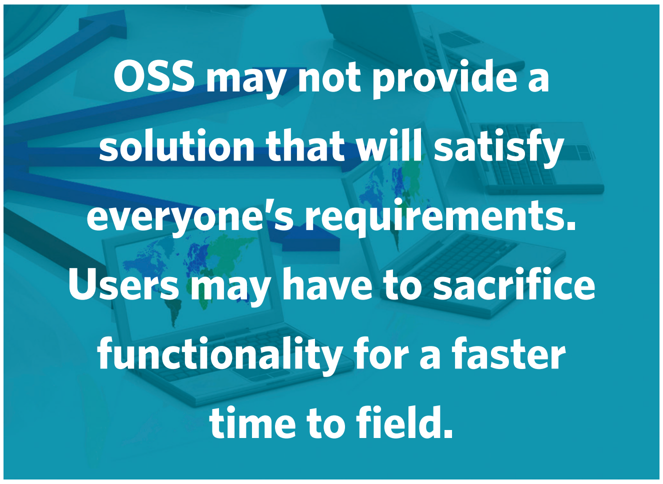
Open source software is generally thought to be free as in it has no costs. Though that is true in most cases, generally the term "free" is used in reference to the liberty of interested parties to freely distribute the source code. That is an important aspect to keep in mind when considering the use of OSS—there may be a cost.

Like proprietary software, OSS comes with licenses such as the GNU or Apache license. This article does not cover the licensing associated with OSS; however, it is important that the proper legal representative reviews the license prior to making the final decision. This assures that the manner in which interested parties intend to use the OSS is in accordance with the license.

Is OSS an Open System?

There is no direct correlation between an open system and OSS. Open source specifies that the human-readable source code of the application is available. In contrast, an open system, as defined by the Open Systems Joint Task Force, is specified as "A system that employs modular design, uses widely supported and consensus based standards for its key interfaces, and has been subjected to successful validation and verification tests to ensure the openness of its key interfaces. "

The question as to whether OSS meets the definition of an open system must be addressed per DoD Directive


OSS may not provide a solution that will satisfy everyone's requirements. Users may have to sacrifice functionality for a faster time to field.

5000.01: "A modular, open-systems approach shall be employed, where feasible." Because there are generally many contributors to open source projects, they tend to have a modular design; however, this is not always the case. Open Office has 450,000 members that have joined the project, so enforcing a modular design is paramount for continued success. Without a modular design, it would be extremely difficult to modify the source code of such a large application with so many contributors.

Another part of the open system definition is using consensus-based standards for key interfaces; this is also referred to as using open standards. Open standards play a critical role in our systems with modifiability, maintainability, and increased competition. Open standards have no direct correlation to OSS. Though most OSS projects use open standards, it is not required. Each OSS project must be assessed individually to determine if it is, indeed, an open system.

Are the Releases Controlled?

As with most software, OSS has multiple versions, releases, and security updates of which one's program is not in control. The need for life cycle configuration management is vital in ensuring system compatibility. A strategy needs to be developed to determine how one's program will handle periodic releases of the OSS software. Depending on the software, each release may require configuration, interface and installation, or system changes to remain compatible with the rest of the system.

What is the Maturity of the Open Source Community?

Similar to a standard commercial company, the maturity and size of the open source community can vary greatly. Open source projects can be started by a single developer who has made its source code available and gained additional support as the project grew, or by corporations who fund and assist in the development of the project. Open Office, an open source office suite, is sponsored by Sun® Microsystems and has other corporate contributors such as Google® and IBM®. The Open Office project contains 30,000 source files and 9 million lines of primarily C++ code, according to the Open Office website, and it contains many of the features included in Microsoft Office.

Many factors affect the maturity of the open source community supporting the project. Navica® has developed an Open Source Maturity Model®, which is freely available and will assist in the assessment of the open source project. The Open Source Maturity Model provides a variety of templates to assess different areas of the open source project such as documentation, integration, product software, professional services, technical support, and training. Those items are then further decomposed to help assess each area of the open source project.

Do You Need to Modify the Source Code?

The major difference between proprietary software and OSS is the ability to view, modify, and distribute the application source code. Code modification may lead to some undesired effects on the life cycle of the system. Modifying the source code would force the program to keep a private copy that is different from the open source project's repository. That may work without issue for the initial release, but remember, just like proprietary software, OSS periodically releases new versions, patches, and upgrades. Once one breaks off from the primary project, he or she is now responsible for any upgrades and associated testing as the releases may not be compatible with the modified version.

Code modification may not be as easy as one might think. Take the Open Office project mentioned previously. If someone required a code modification and provided the development team with 9 million lines of code, a seemingly trivial modification may turn out to be a daunting task. Unfamiliarity with the application or programming language may cause additional complications. Most OSS uses a modular design so it can be easier to locate the code segment for which the modification is needed; however, the effects on the application may still be unknown.

One possibility is to make the modifications to the source code and submit the update to the OSS project's committee for review and possible incorporation within the next software release. If accepted, the update would go through the project's revision, testing, and review process during subsequent releases, and one would no longer need the old version of the software. Similar to most commercial software, the open source community does what is best for the community and not one's specific program. Therefore, there is no guarantee one's changes will be included in the next software baseline. As with any software application, when new functionality is added, the project is now responsible for maintenance, testing, and bug fixes for the added piece of functionality.

While modifications provide an added level of complexity, OSS does provide several alternatives over commercial software. One alternative may be deciding there is only a need to use a portion of the source code within the project. If the OSS is modular in design, it may be easy to extract only the functionality needed to incorporate into the application. That may be the best option if only a small piece of the OSS functionality is required. As with proprietary software, there is a point where "too much of a good thing" can turn bad. If one takes several pieces of different systems and includes them in his system, the system may become difficult to maintain, especially when each addition is in a different programming language, contains different interfaces, and may require additional dependencies. This can be exemplified by using a car analogy. Consider buying a Chevy Camaro but realizing that it will require the engine in the Ford Mustang and the electronics of the Audi A4. After integrating the required functionality of the other automobiles, the owner would have a system that met all of his requirements. However, if the vehicle needed maintenance, the owner would no longer be able to take it back to the Chevy dealership because a modification to the electronics system may adversely affect the engine because the components were not initially design to work together. In addition, if Audi releases an electronics upgrade, the owner may be unable to use the new software due to compatibility issues with the nonstandard engine.

Is OSS the Full Solution?

As with most proprietary products, OSS may not provide a solution that will satisfy everyone's requirements. Users may have to sacrifice functionality for a faster time to field. Gen. David Petraeus, commander of U.S. Central Command, recently said in an interview, "Never underestimate how important speed is." Additionally, he pointed out that in most cases, the soldiers are willing to accept an 80 percent solution. This is where constant user involvement is imperative in order to help make an informed decision. The user decides if less functionality provided sooner outweighs the time needed to develop the functionality from the ground up.

Conversely, OSS comes with a variety of features and could include many more features than are required by one's program. This inundation of extra features may require additional training, testing, and/or information assurance assessments to use the software in an operational environment. Removal of those features is also an option, but one must remember the risks mentioned in the modification section.

Does OSS Offer Maintenance and Support?

OSS may also contain a maintenance and support element that is available for a cost. MySQL offers an enterprise package that includes the software, support, and additional monitoring tools. Depending upon the needs of the program, one may consider a support package in which

SoftwareForge hosts open source and community software projects within the DoD. If public availability it not an option, SoftwareForge may be a more secure alternative.

the cost would need to be added into the life cycle cost of the system.

Overall Evaluation of OSS

If one chooses to modify the source code and keep his own version, OSS can easily morph into government off-the-shelf software, losing most of the value of leveraging from the OSS community. At that point, the program becomes responsible for having developers available for maintenance and support. One may also find himself maintaining a great deal more features than what is required for the program. Most OSS projects make the executable (installer) available for download. If one were to only download the executable, he will be left with what is essentially a proprietary product but with the added benefit of having access to the source code. Modifying the source code may be a researcher's best option as long as he is prepared for the possible future consequences.

The items identified in this article are only a few of the considerations for evaluating OSS for use within a program. Other factors that may need consideration are security, prerequisites, reliability, and performance. The DAU Best Practices Clearinghouse (<https://bpch.dau.mil>) contains a forum to enable the sharing of best practices when evaluating OSS throughout DoD.

Remember, the open source community is available because projects make their source code available. Making someone's code available may allow for external reviews and could improve code quality. The Defense Information Systems Agency has developed an online open source repository at <www.forge.mil> called SoftwareForge. SoftwareForge hosts open source and community software projects within the DoD. If public availability it not an option, SoftwareForge may be a more secure alternative.

The author welcomes comments and questions and can be contacted at **matthew.kennedy@dau.mil.**