Cracking the Code: Contracting for Open Source Software

Randy Saunders
Johns Hopkins University Applied Physics Lab
Laurel, MD
Randy.Saunders@jhuapl.edu

Dr. Gary Allen
Joint Training Integration & Evaluation Center
Orlando, FL
Gary.Allen@us.army.mil

ABSTRACT

Open source software licenses provide government programs with excellent opportunities to reuse software without paying license fees or incurring annual software maintenance fees. From Linux to simulation tools, many programs take advantage of existing open source licenses. Simulation developers have long been free to spend their investment money on a piece of software they give away through open source. Contractors don't have that capability with software developed with government funds, unless they receive special direction. The government, on the whole, endorses the use open source software, but is often hindered by conflicting policies on how to proceed.

What happens when you can't find a piece of open source software to fit your requirements? How can you avoid intellectual property claims that force you to use the original developers for future software modifications? You may be able to solve these problems by paying for a simulation developer to produce an open source software product. The product is made widely available, so any competent developer can enhance it. If the community benefits enough to maintain it, you might be able share your support costs.

This paper describes how JHU/APL was directed to produce a tool as open source software, including the contractual language needed to specify required license terms and accommodate DFARS 252.204-7000, Disclosure of Information. The license selection results are provided and we describe the process for establishing the open source repository. The tool itself was produced and is available as free, open source software today.

ABOUT THE AUTHORS

Randy Saunders is a Principal Staff Engineer at the Johns Hopkins University Applied Physics Laboratory. He has over 30 years of experience in the design, implementation, and integration of high-fidelity simulations for military and business customers. He received his B.S. and M.S. degrees in Engineering from Harvey Mudd College in 1980 and an M. S. in Computer Science from the University of Southern California in 1985. Mr. Saunders has been active in distributed simulation standardization since the first DIS Workshop, in both DIS and HLA standards committees, and is presently a chair of the HLA Evolved Product Support Group. He joined APL in 2001.

Dr. Gary Allen has worked in various aspects of modeling and simulation for the past 30 years. He was a member of the team that founded the Training Simulation Center for I Corps at Ft Lewis Washington, Director of the Simulation Training Branch at the US Army Intelligence Center and School, Ft Huachuca, and Project Director for the TACSIM Intelligence Simulation and part of the design group that initiated the Aggregate Level Simulation Protocol (ALSP). Dr. Allen spent 29 years on active duty in the US Army finishing his career as Commander of the International Technology Center at the US Embassy, Germany. His DoD Civilian Career began in 2002 as the US Army Liaison Officer to the German Military Research and Development Agency in Koblenz, Germany until 2009. Currently he is at PEO STRI as the Project Manager for the DoD High Level Task, 'Live, Virtual, Constructive Architecture Roadmap Implementation' project. 1978 MS - Telecommunications Systems Management, School of Engineering, University of Colorado 1989 PhD – Instructional Technology, University of Kansas 1999 – Graduate, US Army War College Dr. Allen is a graduate of the Defense Language Institute (German), a member of the Phi Kappa Phi National Honor Society, and a DoD Acquisition Corps Level III Certified PM.

Cracking the Code: Contracting for Open Source Software

Randy Saunders
Johns Hopkins University Applied Physics Lab
Laurel, MD
Randy.Saunders@jhuapl.edu

Dr. Gary Allen
Joint Training Integration & Evaluation Center
Orlando, FL
Gary.Allen@us.army.mil

WHAT IS OPEN SOURCE SOFTWARE?

Open Source Software (OSS) provides a different approach to software development, distribution, and support. Software developers provide users with more useful implementation details (the source code for the software) and more intellectual property rights (as defined in an open source license). In exchange, users become software collaborators, responsible for reporting and repairing software problems. In the modeling and simulation (M&S), training, testing, and education fields this distribution of rights and responsibilities enhances reuse and leads to more cost efficient procurement than traditional "hands-off" software contracting.

Open Source License Characteristics

Software licenses recognized by the Open Source Initiative[1] share common characteristics including:

- Redistribution without royalties or fees
- Source code to fully disclose the design and implementation of the software
- Modification and derived works are allowed

OSS isn't a "something for nothing" scheme. While these license terms prohibit per-user, per-machine, and per-year fees; software producers still expect to get paid, profitably, for their efforts. What OSS removes from the business equation is the "low upfront price to lock-in a long term revenue stream" pricing tactic where software vendors bid unreasonably low prices for development in hopes of charging more for future support/upgrades where their incumbency represents a barrier to entry. OSS similarly impacts programs looking for a low cost initial capability where long-term support costs are funded separately.

Open Source Benefits

The primary benefits of OSS are: 1) agile response to user needs; and 2) freedom to contract with a broader group of software collaborators.

Traditional software development contracts define a set of requirements and measure accomplishment against those requirements. This approach works best where software is needed to perform a well-understood, unchanging, function for a long period of time. Today's warfighter, on the other hand, faces constantly changing functional assignments and often utilizes equipment in ways never imagined by its designers.

The result has been poor user satisfaction with some software systems. The user needs a simulation to do things it was never designed to do, and may find it less adaptable than the system it simulates. Managing software cost requires simplifying the implementation of a simulation; you can't afford to simulate every aspect of the system. When usage changes, the shortcuts appear as problems.

Because the traditional approach ignores this change, nothing is done to facilitate it. A new software task faces a difficult choice: returning to the original developer for unique insight into the software puts the buyer in a weak position to negotiate a price; or competing the change when the competitors don't have the access to study the existing software minimizes the reuse savings. The buyer pays more for software changes either way.

OSS benefits the warfighter by enabling collaboration among developers. Without the lock-in incentive, initial developer can focus on reuse and optimizing the implementation. When requirements shift, the developer best able to make the desired change can step in to make the needed corrections. In a more agile development environment, developer goals are better aligned to user goals; the developer no longer benefits from duplication of functionality across several programs. For example, if the manufacturer of a new weapon system builds OSS models for analysis and testing support then those models could freely be reused within a training simulation. This provides double benefit; the trainer usage maintains the models so that they are available for follow-on analysis, and there is no need to redundantly model the weapon system for inclusion in the training system.

CONTRACTING FOR OSS

Work on an OSS development occurs by defining an open source license to be applied and an open collaboration space. Developers are paid to implement specific capabilities and integrate them. Users can download and use the software freely, sharing the costs of development and maintenance. This sharing motivates users to support fewer programs, reducing duplication, and to welcome additional users as a support cost reduction rather than a potential conflict.

Contracting Officer Concerns

Typically the first hurdle in providing for acquisition initiatives that are outside the 'usual' way of doing business is to set regulations and policies in place that clearly establish the legal permissions needed to allow for an acquisition to take place. This situation is certainly true in the case of the US Governments acquisition and use of OSS. Because of a number of popular misconceptions regarding the use of OSS the process of getting the various regulations and policies in place has taken some time. These misconceptions include licensing issues, indemnification, forced distribution of source code, threats to national security, and reliability just to name a few. Regardless, the ability to acquire OSS is well covered by regulations and policies. As far as DoD is concerned OSS comes under the heading of "commercial computer software" products. The relevant authorizing conduits are:

- United States Code 41 USC 403[2]
- Federal Acquisition Regulation (FAR) 2.101(b), 12.000, 12.101[3]
- Defense Federal Acquisition Regulation (DFAR) 212.212, 252.227-7014 (a) (1)[4]

These various regulations do, however, have some ambiguity and are subject to interpretation or in some cases can be misconstrued. This has led to DoD issuing policies to provide additional guidance in the acquisition of OSS. Those DoD policies include a memorandum issued in May 2003[5], "Open Source Software (OSS) in the Department of Defense" which covers issues relating to OSS licensing, compliance with the policies that cover Commercial off the Shelf Software (COTS), Government off the Shelf Software (GOTS), and Information Assurance Guidelines as covered in DoD Instructions 8500.1/8500.2[6]. An additional policy memorandum was issued in October 2009, "Clarifying Guidance Regarding Open Source Software (OSS)"[7] which recognizes various misconceptions that exist and hampered acquisition efforts to acquire OSS. The memorandum provides an appendix that provides clarifying guidance related to those issues.

While the previous regulations and policies provide the authority for contracting officers to support the acquisition of OSS, project management needs to be sensitive to the concerns of their serving contracting officials and be sure to address those concerns as the acquisition team puts together the Request for Proposal (RFP) or solicitation. Therefore it may be incumbent upon the program management side to reinforce the following points when working with their contracting officer:

- Commercial products are preferred and OSS is considered to fall into that category
- DoDD 8320.02 defines source code as 'data' and data must be shared within DoD
- It is appropriate to release DoD developed software to the public under certain conditions
- OSS needs to be part of market research
- The need for DoD to develop and update capabilities faster is an OSS advantage
- Through the use of OSS solutions projects have the possibility to attain increased functionality, quality, and flexibility
- OSS can reduce development time and costs

Another area of concern for contracting officials is the ability to hold providers responsible for the products they deliver. In other words, 'whom do I sue' if the Government does not get what they paid for. In the case of OSS there are few companies willing to offer indemnification for their OSS products and most will typically refuse to be held legally accountable beyond what is specified in the OSS licensing agreement[8]. There are few alternatives to this situation making it an important point that must be discussed when looking at OSS solutions. One guard the contracting official can take is to insist on the delivery of source code and any other related materials it has rights to before making payment.

Contractor Concerns

The contractor's first reaction to a different way of doing business depends on their past success. The successful incumbent believes "If it's not broke, don't fix it.", expecting past success would indicate future success if the business process remains the same, all other things being equal. The emerging challenger believes things need to change, for the same reason. They are both wrong. Things are not "equal" to what the industry has seen for the past decade because the money simply isn't available.

Eliminating the "software lock-in" incentive doesn't mean all software developers will abandon their differences and join together in a happy campfire. Engineers are competitive, and excellence in software emerges from that competitiveness. OSS simply provides a mechanism to focus that competitive spirit on improving the common solution, rather than matching advances in redundant solutions. The debate should shift from "which software has added the most useful features" to "which features would be the most useful to add to the common software".

Contractors have two kinds of concern with the transition to OSS:

- Without lock-in, will my company be successful in a competitive marketplace?
- Will collaboration with other developers get me in trouble for something they did?
- Will my business make money with OSS?

Making a company more competitive is necessary for the long-term success of their business, so most attention will be on the latter concern.

Two legal concerns, beyond those mentioned in the previous section, are important to contractors:

- DFAR 252.204-7000[4], Disclosure of Information, requires Contracting Officer approval before information can be disclosed.
- 22 U.S.C. 2778[2] of the Arms Export Control Act (AECA) controls disclosure of software (among other things) outside the US.

Internet collaboration is virtually synonymous with disclosure of information, often without regard for US laws, and boundary-less international participation. No contractor should be willing to face the stiff penalties embodied in these laws for the acts of third parties with which they have no contractual relationship.

Nobody benefits if software developers are not turning a profit. The companies stop doing the work, and the marketplace collapses. When OSS software can be copied and used for new applications, how does a company make money? The companies make money the same way they do today, by adding value. Many M&S software projects are already built under the terms of DFARS 252.227-7014[4], Rights in Noncommercial Computer Software & Documentation, where the government has unlimited rights. The software vendor does not get paid to make another copy of the software. The company gets paid to take a piece of existing software and enhance it. The same profit making approach applies to OSS. The company who can best make an enhancement gets paid to make the enhancement to the OSS project.

FEAT EDITOR EXPERIMENT

The Live/Virtual/Constructive Architecture Roadmap (LVCAR) Implementation study of business models[9] recommended an experiment, as an exemplar for others working to pursue OSS development. The Federation Engineering Agreements Template (FEAT)[10] work developed an XML schema for representing federation agreements, but some users found XML difficult to work with directly. A tool to facilitate use of the schema to document agreements provided an opportunity to begin a software development effort without the burden of pre-existing intellectual property issues. While work continued at the Simulation Interoperability Standards Organization (SISO) to standardize the XML, a pilot OSS project began work on a tool.

Contract Language Overview

The key to dealing with the issues raised so far is clear direction to the software developer to use an OSS approach. As in all government contracting, the details are in the details. The language that follows is the actual language used for the FEAT Editor experiment, none of which should be construed as legal advice or a guarantee that any particular Government Contracting Officer will be completely happy with the language should you use it. The symbol "{software}" is used in place of the actual software description to shorten this paper.

Develop Software

In order to have open source software development, the contract or task assignment must direct software to be developed. In the FEAT Editor example, the following clause was used:

> Develop a prototype {software}, as an open source software application, to assist in implementation of the template by users.

Special Requirement 1 = Open Source Development

In order to direct the development of OSS, it is important to specify two characteristics of the implementation strategy needed:

1. Software Development Model
2. Open Source License Attributes

The identification of a software development model defines the degree of involvement allowed to contributors outside the contractor. If others are only allowed to access the software, the contractor has less involvement with contributors but contributors are less likely to make a significant investment. Conversely, full and active participation by outsiders causes administration and consumes some contractor time and effort.

For the FEAT Editor example, wider participation was seen as a benefit that more than offset the admin costs, so the following collaborative language was used:

> Open Source (OS) Software Development Model
>
> Using an OS software development model described below, JHU/APL shall collaborate with multiple other contributors/companies, including non-U.S. contributors (collectively, "contributors"), to develop the {software}. JHU/APL will manage a centralized repository, such as a common server accessible to all contributors, for hosting software developed by the contributors. Preferably, the contributors will develop and enhance OS software under a set of compatible OS licenses, or preferably, a single OS license, and then share their software with each other, and the U.S. government, via the repository.
>
> The contributors will populate the OS repository with their respective OS software (including their source code). Each contributor will have unrestricted access to the OS software of the other contributors ("other OS software") and may further develop the other OS software under the terms of the common license, e.g., LGPL. Each contributor may then release such further developed OS software into the OSS repository under the common license. In this manner, the contributors will develop and share OS software with each other in an open manner through the OS repository and under the common license.

The specific open source license used impacts not only what rights contributors have, but it also influences who will want to be a contributor. In addition to the general characteristics discussed above, some open sources licenses have provisions called "copyleft". Strong copyleft licenses, such as the GNU General Public License (GPL), require that any contributor who uses the software make all software components available under the GPL license. Contributors who want to build the open source software into their products will not find this license acceptable. Weak copyleft licenses, such as the GNU Lesser General Public License (LGPL) require contributors to provide source to changes, but not to their own additions. Permissive, non-copyleft, licenses such as the Berkeley Software Distribution (BSD) license permit contributors to sell their modified versions of the software. These permissive licenses are not very desirable for government funded open source projects because they could easily require the government to pay again for updated versions of the software.

Contract language might specific a particular open source license to be used, or it might direct the contractor to select one consistent with the government's needs. In the FEAT example, the latter approach was used with the following language:

> Preferably, the common OS license shall have weak copyleft attributes that:
> (i) cause modifications of the OS software to retain the OS license due to the effect of copyleft, and thus require re-release of source code into the OS library (i.e., repository), but
> (ii) permit proprietary software to retain its proprietary status when linked with the OS software.
> LGPL is one candidate OS license with such attributes.

Special Requirement 2 = Export

Because the Internet extends beyond the U.S., publication of information on the Internet must be done in compliance with applicable export laws. For software with only military applications, the jurisdiction over export control falls to the U.S. Department of State and the contractor must be tasked to get appropriate export approval. For non-military software, export jurisdiction falls to the Commerce department. Software not on their control list can be exported without a specific export approval. In either case, all other export laws such as the prohibition of export to embargoed countries must be followed. In the FEAT Editor case, the following language was used to flow down these requirements:

> The {software} (including the OS software) developed under the task is not specifically designed or modified for a military application. Therefore, it is controlled by the Commerce Department. As the software is not specifically listed on the Commerce Control List, it is classified "EAR99" and can be

> exported to any country except to an embargoed country, to an end-user of concern or in support of a prohibited end-use. (See EAR Parts 744 (Control Policy: End-User and End-Use Based) and 746 (Embargoes and Other Special Controls).

Special Requirement 3 = Government Licenses

Normal contract language grants "Government Purpose Rights" to for all software developed by contractors. When the contractor has previously developed software with more limited rights, those restrictions are disclosed before the contract is awarded. The common OS licenses don't implement this notion of government purpose rights, and generally try to explicitly declare government and private users equal. Specific language is needed to accept rights restrictions on unwritten software, and in the FEAT Editor example, the language used was:

> The {software} (including the OS software) will be accessible/delivered to the U.S. Government. The U.S. government will acquire license rights in the {software} in accordance with the relevant OS license, consistent with DFARS 227-7202 Commercial computer software and commercial computer software documentation. The development of OS software under the task is consistent with the DoD Memorandum, "Clarifying Guidance Regarding Open Source Software (OSS)" issued October 16, 2009.

Special Requirement 4 = Disclosure of Information

As previously discussed, the disclosure of information clause need not be completely waived in order to direct participation in an OS project. Only the information released as part of the OS collaboration: bug tracking lists, release notes, documentation, and the source code itself should be covered by the disclosure provisions. This I/ITSEC paper, for example, worked through the normal public affairs office release process as it was not in the narrow collaboration needed for OS development. The FEAT example language was:

> JHU/APL is permitted to share information, including software and related technical data, developed by JHU/APL under the task with all of the other third party contributors to the {software} permitted to access the repository.

EXECUTING A CONTRACT FOR OSS

Export Assessment

Software must be evaluated for exportability before it can be developed as OSS. The US Department of State operates a Commodity Jurisdiction (CJ) request system to verify the exportability of all items. In 2010, the system became completely electronic, significantly reducing wait times.

Military utility does not automatically make software export controlled. Mechanisms like parameterization of military-relevant information, frequently used to separate unclassified algorithms from classified data coefficients, can also be applied to improve export approval likelihood.

Hosting Approaches

Even exportable products cannot be exported to embargoed countries. Choosing a hosting solution for an OSS project must comply with this legal requirement. Any US-based hosting provider must have a mechanism for supporting compliance, and a spectrum of levels of assurance is available. For higher assurance, Forge.mil operates an OSS project hosting service within the US DoD. While Forge.mil is optimized for CAC users, access is available to all DoD contractors.

Development of FEAT Editor

For the FEAT Editor experiment, the export control assessment was EAR99, the non-military designation, allowing export to all but embargoed countries. The international nature of SISO, and thereby the prospect of more collaborators if international participation was encouraged, led to the use of Sourceforge.net as the hosting approach. Feel free to join the project and make contributions to this OSS development.

EXPERIMENT RESULTS

Successful Contracting

After a survey of a number of recent articles and discussions with contracting officials the general consensus seems to be to start early in the requirements and solicitation process and identify that OSS will be considered. This, in turn, will bring any issues the contracting officer may have to the forefront. This is especially true in cases where the responsible contracting officers may have limited experience with software development. In a recent article from Fierce Government IT a DoD representative working with contracting officials that they had to take a different tact; "For us to continue to progress it was a mentality of don't take the first 'no' you get. We began asking a lot of 'why not's?'"[11]

Licensing is probably the one area that is most often misunderstood due to unclear guidance in DFAR 227.7202-3, Rights in Commercial Computer Software or Commercial Computer Software Documentation[4]. This DFAR stipulates that the Government shall have only the rights that are specified in the license obtained in the contract. If there is a Government requirement for rights outside of the license available to the general public then that is a point of negotiation with the vendor. The negotiation issue can cause special problems for OSS[12]. Those problems are:
Inability to Negotiate – intellectual property rights in OSS generally are not a topic for negotiation. The rights provided for in the licenses that govern OSS ultimately define the government's rights.
Viral Licensing – this is where OSS source code that is modified inherits the same license agreement as the original code. If the Government wants to modify OSS or merge it with 'classified code' and restrict the distribution then this is a violation of the license the original code was procured under. This means that the Government must be willing to distribute modified code as freely as the terms of the original license provides which may eliminate an OSS solution. Currently there are over seventy licensing regimes governing OSS that have gone through some licensing approval process[8].

In order to provide for success early in the contracting process the DoD Open Source Contract Guidebook[12] cautions that the Government needs to implicitly understand:

- What the open source software is and the licensing constraints for the open source software
- How the open source software will be used within the system being procured
- Whether it is likely the OSS will need to be modified and/or distributed over the lifecycle of the system
- The license impacts of non-open source computer software used in modifying OSS
- OSS is automatically licensed to a user on nonnegotiable terms

While there certainly are inherent risks in contracting for OSS solutions they are no more daunting than with other contracts involving software development and delivery. The key to success is an acquisition team that understands where the risks lie and develops an acquisition strategy that addresses those risks.

Successful Execution

The FEAT Editor was established on Sourceforge.net and software has been developed. Linkage to the SISO product development group[10] provides information about the tool to the broader community.

Exemplar for Future OSS Contracts

OSS projects are gaining in popularity on a global scale. In a 2011 survey conducted by 'FuseSource', an international IT organization that advocates the use of open architectures, there were clear indications that the development and use of OSS was on the rise in a significant manner[13]. An informal review of Sourceforge.net indicates that there are approximately 20,000 software projects involving over 70,000 developers worldwide[14]. This provides good evidence that the benefits of OSS development are being realized and DoD is certainly among that group.

Not to be excluded from taking advantage of OSS the LVCAR-I High Level Task purposely planned to use OSS in areas where it made sense to do so. Given that we are working on DoD Enterprise wide interoperability tasks we needed the ability to easily distribute the developed software products, which led us early on to pursue the use of OSS. Following the recommendation that successful contracting in OSS is best accomplished during the project requirements process we included its use in the project's implementation plans. The next step was to include the requirement in the Statement of Work (SOW). By including the requirement in the SOW the performer then had a government requirement to fulfill thus eliminating the need to provide any additional justification for the development effort. The following is offered by way of a short background on the FEAT:

- Objective: Develop an architecture-independent template for establishing federation agreements that can be supplemented by potential architecture-specific extensions
    - All LVC developments need to establish federation agreements, but there is no standard/convention for the structure or content
    - Projects currently develop their own mechanisms for such agreements,

thus reducing the opportunity for reuse

- A common template, with flexibility for architecture-specific extensions, facilitates some uniformity in how agreements are documented, facilitating reuse both within and across user communities, and enabling tool support

To meet the stated objective and provide the user with a machine readable capability the tool is being developed using XML schemas that in the end provide not only a sophisticated electronic 'to do' list but provides a way to share information, have an accurate record of what was done, and in following good OSS practices provide the greater DoD Modeling and simulation community with a capability to readily update the tool and share those updates. As far as the SOW is concerned we simply stated that OSS would be used as shown here.

> "Federation Agreements Template. The contractor shall continue to lead the development of the Federation Agreements Template standard under the auspices of the SISO Federation Engineering Agreements Template (FEAT) PDG. In parallel with the PDG effort, the contractor shall improve the robustness of the prototype Federation Agreements Template tool in order to increase adoption of the template and to support other LVC Common Capabilities goals. The Federation Agreements Template tool shall be an open-source product."

The FEAT Editor provides a good example of the use of OSS but is only one of many. The base point is that using OSS in DoD is doable provided that the key aspects/concerns of contracting are addressed and that the requirement has a good fit to the advantages offered by OSS.

REFERENCES

[1] www.opensource.org
[2] US Code, available online at http://www.gpo.gov/fdsys/
[3] Federal Acquisition Regulation, https://www.acquisition.gov/far/
[4] Defense Federal Acquisition Regulation, http://www.acq.osd.mil/dpap/dars/dfarspgi/current/index.html
[5] Department of Defense Memorandum, 28 MAY 2003, SUBJECT: Open Source Software (OSS) in the Department of Defense (DoD), http://www.terrybollinger.com/stenbitmemo/stenbitmemo_pdf.pdf
[6] DoD Instruction 8500.1 and 8500.2, available through DAU at https://acc.dau.mil/CommunityBrowser.aspx?id=180761
[7] Department of Defense Memorandum, 16 October 2009, SUBJECT: Clarifying Guidance Regarding Open Source Software (OSS), http://dodcio.defense.gov/Portals/0/Documents/OSSFAQ/2009OSS.pdf
[8] Oracle White Paper, March 2009, The Department of Defense and Open Source Software, http://oss.oracle.com/
[9] LVCAR Reusable Tools Implementation – report http://www.msco.mil/documents/LVCAR-I%20FY09%20Reusable_Tools_Implem_Plan.pdf
[10] Federation Engineering Agreements Template (FEAT) Product Development Group (PDG), http://www.sisostds.org/StandardsActivities/DevelopmentGroups/FEATPDGFederationEngineeringAgreements.aspx
[11] Walker, Molly Bernhart, 4 FEB 2011, "Open source in government still difficult, say conference attendees", Fierce Government IT, http://www.fiercegovernmentit.com/
[12] OSA Contract Guidebook v.0.1, 15 DEC 2011 DRAFT, Appendix 5: Open Source Software, https://acc.dau.mil/adl/en-US/489360/file/61990/OSA%20Contract%20Guidebook%20v.0.1%208.5x11_12-15-2011_cs.pdf
[13] FuseSource, JUL 2011, FuseSource Survey on Open Source: How enterprises are using open source software, http://fusesource.com/collateral/145
[14] Lechner, David and Kaiser, Harold, Defense Acquisition Review Journal, vol 12 no 4, Dec 2005-Mar 2006, p375-391, http://www.dau.mil/pubscats/PubsCats/AR%20Journal/Lechner.pdf