NPS-AM-10-037

EXCERPT FROM THE

PROCEEDINGS

OF THE

SEVENTH ANNUAL ACQUISITION RESEARCH SYMPOSIUM WEDNESDAY SESSIONS VOLUME I

Acquisition Research

Creating Synergy for Informed Change

May 12 - 13, 2010

Published: 30 April 2010

Prepared for: Naval Postgraduate School, Monterey, California 93943

The research presented at the symposium was supported by the Acquisition Chair of the Graduate School of Business & Public Policy at the Naval Postgraduate School.

To request Defense Acquisition Research or to become a research sponsor, please contact:

NPS Acquisition Research Program
Attn: James B. Greene, RADM, USN, (Ret.)
Acquisition Chair
Graduate School of Business and Public Policy
Naval Postgraduate School
555 Dyer Road, Room 332
Monterey, CA 93943-5103
Tel: (831) 656-2092
Fax: (831) 656-2253
E-mail: jbgreene@nps.edu

Copies of the Acquisition Sponsored Research Reports may be printed from our website www.acquisitionresearch.net

On Open and Collaborative Software Development in the DoD

Scott Hissam—Scott Hissam is a Senior Member of the Technical Staff for the Carnegie Mellon Software Engineering Institute, where he conducts research on component-based software engineering, open source software, and multicore. Mr. Hissam is a founding member and secretary of the International Federation for Information Processing (IFIP) Working Group 2.13 on Open Source Software and co-organizer of its annual conference. His publications include two books (*Building Systems from Commercial Components* and *Perspectives on Free and Open Source Software*), papers published in international journals, and numerous technical reports. He has a BS in Computer Science from West Virginia University.

Scott A. Hissam
Carnegie Mellon Software Engineering Institute
4500 5th Avenue
Pittsburgh, PA, 15213 USA
+1.412.268.6526
shissam@sei.cmu.edu

Charles B. Weinstock—Charles B. Weinstock is in the Research, Technology, and System Solutions Program at the Software Engineering Institute. His main interest is in dependable computing. For the last several years, he has been developing assurance case technology. He is also active in the open source software community. Previously, Weinstock worked at Tartan Laboratories and SRI International. Weinstock has a PhD in Computer Science, an MS in Industrial Administration (MBA), and a BS in Mathematics, all from Carnegie Mellon. He is a Senior Member of the IEEE and a member of IFIP Working Group 10.4 on Dependable Computing and Fault Tolerance.

Charles B. Weinstock
Carnegie Mellon Software Engineering Institute
4500 5th Avenue
Pittsburgh, PA, 15213 USA
+1.412.268.7719
weinstock@sei.cmu.edu

Len Bass—Len Bass is a Senior Member of the Technical Staff at the Software Engineering Institute. He has authored two award-winning books in software architecture and several other books and papers in various computer science and software engineering areas. He has been a keynote speaker or a distinguished lecturer on six continents. He is currently working on techniques for the methodical design of software architectures, supporting usability through software architecture, and to understand the relationship between software architecture and global software development practices. He has worked in the development of numerous software systems, ranging in a multitude of domains.

Len Bass
Carnegie Mellon Software Engineering Institute
4500 5th Avenue
Pittsburgh, PA, 15213 USA
+1.412.268.6763
ljb@sei.cmu.edu

Abstract

The US Department of Defense (specifically, but not limited to, the DoD CIO's Clarifying Guidance Regarding Open Source Software, DISA's launch of Forge.mil and OSD's Open Technology Development Roadmap Plan) has called for increased use of open source software and the adoption of best practices from the free/open source software (F/OSS) community to foster greater reuse and innovation between programs in the DoD. In our paper, we examine some key aspects of open and collaborative software development inspired by the success of the F/OSS movement as it might manifest itself within the US DoD. This examination is made from two perspectives: the reuse potential among DoD programs sharing software and the incentives, strategies and policies that will be required to foster a culture of collaboration needed to achieve the benefits indicative of F/OSS. Our conclusion is that to achieve predictable and expected reuse, not only are technical infrastructures needed, but also a shift to the business practices in the software development and delivery pattern seen in the traditional acquisition lifecycle is needed. Thus, there is potential to overcome the challenges discussed within this paper to engender a culture of openness and community collaboration to support the DoD mission.

Keywords: Open source software, software engineering, reuse, collaborative development

Introduction

Free and open source software (F/OSS) has been available, in one form or another, for several decades. Successful F/OSS projects benefit from the efforts of a large, usually diverse set of developers. For such projects, the software developed is often as good as or better than the best commercially available software. An even larger community is able to make use of and reap the benefits of this software. The DoD (US Department of Defense) would like to capitalize on this success and adopt an F/OSS model to exploit both reuse among DoD programs and collaboration to improve quality, spark innovation, and reduce time and cost.

The Open Technology Development (OTD) Roadmap Plan prepared for Ms. Sue Payton, Deputy Under Secretary for Defense, Advance Systems and Concepts, identified the following advantages sought from adopting OSS development methodologies (Herz, Lucas & Scott, 2006):

- Encourages software re-use [*sic*],
- Can increase code quality and security,
- Potentially subject to scrutiny by many eyes,
- Decreases vendor lock-in,
- Reduces cost of acquisition,
- Increases customizability, and
- Meritocratic community.

Most recently, Dan Risacher, Office of the Assistant Secretary of Defense (ASD), Networks and Information Integration (NII), was quoted by *Government Computing News* (Jackson, 2008) regarding the benefits of F/OSS as it might apply to defense agencies:

By using open-source software, the services can update their software as soon as a vulnerability is found or an update is needed, rather than wait for the vendor to supply a patch. Open source also promises faster prototyping of systems, and lower barriers to exit. And if a government-written application is released into open source, outside developers could work to fix the problem, lowering maintenance costs of software.

This office is in the process of updating the Stenbit memorandum clarifying the use of F/OSS in DoD programs (Stenbit, 2003).

What is important about these two data points is that they illustrate the level of expectation that is driving the push for the adoption of the F/OSS model of open and collaborative software development in the DoD software community.

This paper explores the idea of adapting the F/OSS model to the DoD software community. While there are a number of other significant concerns mentioned, this paper concentrates on addressing two that are of interest. The first is reasoning how an open and collaborate approach would need to operate in the DoD community, assuming that community was motivated to behave in the same manner as seen in the public F/OSS community. The second focuses on this assumption and reasons as to how to incentivize the DoD community to make use of, and contribute to, such a resource.

The remainder of this paper is laid out as follows: Section 2 looks at the progressive movement towards F/OSS and some of the software reuse repositories (and their challenges) that proceeded today's F/OSS movement. Section 3 takes an abstract view of a project's operation in SourceForge.net as a means for understanding how such resources support the F/OSS community and what they do not do to illustrate a gap that is needed to be filled to support reuse across the DoD community. Section 3 then instantiates this abstract view for use in the DoD to consider the ways in which a DoD-specific resource would compare to that seen in the F/OSS community. Section 4 addresses the prior assumption about behavior expected by the DoD community to consider the incentives necessary to create a healthy and collaborative DoD OSS community. Sections 5 and 6 provide final thoughts on points not yet addressed (perhaps motivating further discussion) and summarize the positions stated in this paper.

The following closely related and relevant topics are beyond the scope of this immediate paper: data rights/licensing issues (commercial, F/OSS, or otherwise); security classifications; various software lifecycle stages beyond IOC (initial operational capability), i.e., pre-RFP (request for proposal) tensions; maintenance of fielded system; field upgrade (new capability); and new systems reusing or proposing to reuse from prior systems.

History of Collaboration and Reuse

There are a number of papers, articles, and publications on the history of F/OSS, some tracing their beginnings to SHARE and the SHARE library in 1955, "to help scientific users grapple with the problems of IBM's first major commercial mainframe" (Gardner, 2005). Others trace to the earlier PACT (Project for the Advancement of Coding Techniques) initiative in 1953, a collaboration between the military and aviation industries (Melahn, 1956; Feller & Fitzgerald, 2001). Feller and Fitzgerald's book provides a nice treatise on the history of F/OSS from these beginnings through the Berkeley Software Distribution, T_EX, the creation of the Free Software Foundation (FSF) and GNU (GNU is Not Unix) and, eventually, to the creation of the Open Source Initiative (OSI). With the advent of

the ARPANET during these emerging beginnings of the modern F/OSS movement, general software repositories began to appear; the most popular included SIMTEL20, originally hosted at MIT (Granoff, 2002), as well as tools to aid in searching these repositories, such as Archie and gopher (Howe, 2009).

With the ever-growing increase in the availability of F/OSS, the benefits of software reuse was also gaining traction within the DoD. In the late 80s (particularly with the DoD's adoption of the Ada programming language) and early 90s, various software reuse efforts within the DoD emerged, including STARS, STARS SCAI, ASSET, CARDS, PRISM, DSRS, ELSA, DSSA ADAGE, and RICC (Department of the United States Air Force [USAF], 1996). Although differences did exist among these repositories with respect to artifact management philosophies, some adopted a generally common theme centered on repositories of reusable software artifacts (code, documentation, etc.) having domain- and/or application-specific classifications, taxonomies, and software architectures all supported by techniques and methods embracing reuse in software development—essentially advocating the concepts that are among the underpinnings of software product lines (SPL) (Clements & Northrop, 2001).

Many of these repositories listed above are no longer in existence, even though their concepts are (in the authors' opinion) sound. Although a case study to completely understand why these efforts ceased would be nice—not the purpose of this paper—we will briefly touch on some of the technical challenges that faced some of the efforts. These include:

- **Quality Arbitration:** The administrative function of deciding what is and what is not included in the repository. This ranges from accepting everything (perhaps resulting in a junk yard or flea market) to a decisive selection (an inventory of few precious selections). Deciding which is the most appropriate is challenging. For the latter, repository customers have higher confidence in artifacts extracted at a higher cost of upfront qualification and an administrative bottleneck in populating the repository. This philosophical difference resulted in two camps: managed and unmanaged repositories.
- **Search and Browse:** At the time of these repositories, free text search and retrieval was a serious resource and computational problem. Free text was not practical; search was a matter of defining a well-crafted database schema, typically relational. There were two approaches. In one, a general purpose schema was defined; in another, domain analysis was used to identify domain specific concepts and terminology. Frakes demonstrated, however, that there was no substantial gain in user search performance obtained by the extra cost and effort of domain analysis (Frakes & Nejmeh, 1987). With time and advances, such free text search capabilities are now more common place (e.g., Google) and no longer presents a major hurdle.
- **Beyond Search and Browse:** Some argued that critiquing domain analysis with respect to retrieval of single reuse items missed the point. Capturing the (sometimes complex) relationships among domain concepts, spanning requirements, algorithms, architecture, code, test, and other artifacts was what was important. The CARDS repository (Wallnau, 1992), for example, used the KL-ONE (Brachman & Schmolze, 1985) semantic network formalism to capture these relations, and use them to support reuse of large-scale domain structures.

Today's work in Web Ontologies also uses a descendant of KL-ONE, and for much the same purpose.

Altogether, this history lesson is worth remembering. In comparison, we believe that the infrastructures supporting the F/OSS community are superior for collaborative development for the projects they service—something that past reuse repositories never imagined. For the larger F/OSS community, these infrastructures are similar to past unmanaged reuse repositories capable of great (seemingly effortless) free text search suitable for opportunistic reuse. We examine this position in more detail below.

Infrastructures for Reuse and Collaboration

There are a number of resources available to the F/OSS community for F/OSS projects including SourceForge.net, RubyForge, JavaForge, Tigris.org, and freshmeat.net, only to name a few. An abstract view of SourceForge.net is created here for the purpose of understanding what such resources commonly do to support the F/OSS community and also what they don't do as a means to illustrate gaps in what is needed to support reuse across the DoD community as well as what would be needed in the DoD to support open and collaborative software development.

SourceForge.net®

SourceForge.net, owned and operated by SourceForge, Inc. (SourceForge, 2009a), is by all accounts one of the most successful source code repositories in the last decade, now boasting over 180,000 projects and nearly 2 million registered users (SourceForge, 2009b). However, simply referring to SourceForge.net as a (software reuse) repository is a great misnomer. Yes, SourceForge.net contains software source code (some of which is reused everyday), but SourceForge.net provides a wealth of other IT-related (hosting and backup) services to the F/OSS community as well as collaborative software engineering and project management tools.

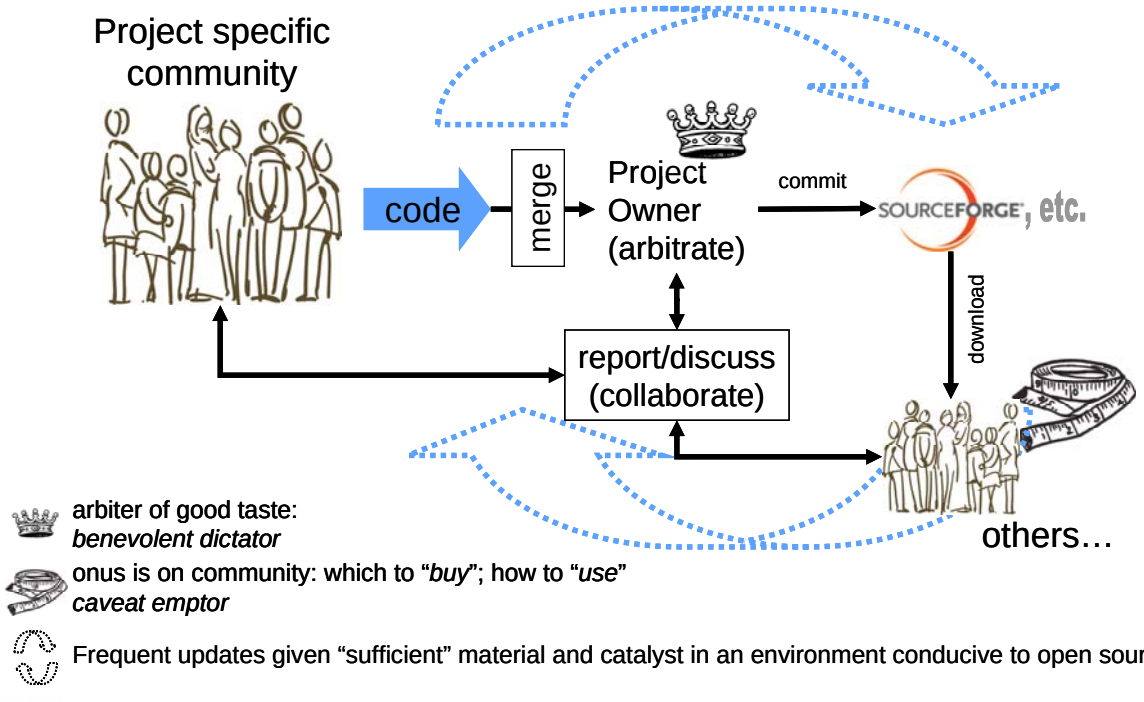

Figure 1. Abstract View of a SourceForge.net Project's Operation

SourceForge.net can best be thought of as a collection of self-contained projects. Each project is administered and owned by a project owner(s) who arbitrates (and delegates) ultimate control over what is committed into the project's code (or artifact) base, what software features are added or removed (over time), and the priorities upon which work progresses. The project's ownership determines the degree of control that is asserted over the project. The project owner is depicted as a crown in **Error! Reference source not found.** as a means to connote the "power" those arbitrators have over the project.

As work progresses, those arbitrators are continuously making collaborative decisions about what is to be done next. For simplicity, the focus for this discussion is on changes offered from the project specific community (on the left of **Error! Reference source not found.**) to the projects artifacts. By balancing their priorities and plans, the arbitrator make decisions on how to merge the interests of this community and the larger F/OSS communities to make changes (and commit those changes) to the artifact base. This churning effect (represented by the cyclic, thick arrows in **Error! Reference source not found.**) is an important and vital aspect of F/OSS collaborative software development. Succinctly, it is this churning and frequent updates (i.e., "release early, release often") to the artifacts that spark innovation through incremental improvements to early and emerging design and source code artifacts given that such updates are open and observable by all in the F/OSS community (Goldman & Gabriel, 2005). This is a continuous, open, and insightful process that is not driven by some external calendar, fiscal boundaries or legal/acquisition milestones.

Lastly, others are free to download software artifacts from the project's repository codebase. This group (in the lower right of **Error! Reference source not found.**) is separated from the project specific community to the left as a means to indicate others[1] that have tangentially "stumbled" upon the project (by whatever means—by search, by reputation, etc.). This group serves a useful purpose in this paper to illustrate another crucial point—that is Eric Raymond's caution in *The Cathedral and the Bazaar*, *caveat emptor*—"let the buyer beware" (Raymond, 2001). This is represented by the large measuring tape in **Error! Reference source not found.**.

Like the earlier users of SIMTEL20, Archie, and gopher, the onus is on this group to determine the degree of fit between artifacts retrieved from the project's codebase and their own needs. One aspect of this determination is partially driven by the need to ascertain if a search actually returned a relevant hit. That is, did the search terms find that which was sought? This is something that was recognized early and many of the DoD software reuse repositories tried to address this with various approaches to classifications and data definitions, for instance ASSET's approach was a faceted classification schema (USAF, 1996; Kempe, 1998) in which CARDS's approach was a domain-specific repository (software for a specific application domain, e.g., command centers). SourceForge.net's classification scheme for projects themselves is limited to broad project categories (for example, Games/Entertainment, Scientific/Engineering, and Security) and subcategories as well as filters allowing other search criteria such as language, operating system, and even licensing. SourceForge.net also provides mechanisms to search across projects (limited to free text searches of project's names and descriptions), to conduct searches within a project

[1] Such individuals may become part of the F/OSS community for a project through a variety of actions, including reporting bug, finding bugs, helping others, porting, contributing ideas, code, etc.

(for example within its documentation, forums, bugs, mailing lists, and configured download packages), and find published files (but not within CVS or SVN—two popular version control systems).

Another important aspect is determining the quality of the artifacts found. If quality is assumed by reputation (e.g., Apache, MySQL, and a host of other reputable F/OSS offerings), this may be no more difficult than in the past with the reputable software of that era (e.g., wuftpd, X, and many of the popular GNU offerings). However, putting reputation aside, quality of the software artifact is at the sole discretion of the project owner—and this has to be discovered in effort expended by the "buyer" through learning, inspection, trial, and testing.

Perhaps the most important aspect is determining if the artifact can actually be reused in the context of the "buyer's" need. The software found may be relevant, and it may be of high quality (by reputation), but may be architected and designed with assumptions that are inconsistent with the context in which it is intended to be reused. One example the authors experienced was to discover that a highly relevant and reputable MP3 encoder/decoder library could not be reused due to the fact that the decoder was implemented in a manner that was not thread safe, even though the encoder portion was. This resulted in an architectural mismatch that prevented reuse in this case. The CARDS and STARS SCAI (USAF, 1996) were some of the earliest DoD software reuse repositories that recognized the need to minimize this mismatch by adopting architecture-centric approaches as a means for qualifying software for reuse within a specific domain.

To summarize key points taken from this abstract view:

- These F/OSS resources (such as SourceForge.net) are for IT-related services housing F/OSS projects and their artifacts with facilities supporting open and collaborative development.
- Project artifacts themselves are managed by a project owner(s) having sole arbitration over the entire project.
- Artifacts are frequently updated and churned over by the F/OSS community, resulting in better quality and innovation.
- It is up to others expending real effort to find, inspect, and assess project artifacts for reuse within their context.

DoDSF

The idea of creating a "SourceForge.net" within the US Government or US Department of Defense, i.e., a "SourceForge.mil" was not invented by us. We credit Schaefer (2005) for the name. Furthermore, the OTD Roadmap called for "an internal DoD collaborative code repository" (Herz et al., 2006). So rather than conflate our analysis with any intent others may have with this idea (either in the past, present or future), we instantiate our thinking by using the term "DoDSF" (a DoD SourceForge).

Like SourceForge.net, DoDSF could also support the IT-related (hosting and backup) services to the DoD community as well as the collaborative software engineering and project

management tools, but cast in the setting of a DoD program acquisition.[2] Using **Error! Reference source not found.** as a basis for DoDSF, **Error! Reference source not found.** illustrates a number of similarities and differences that can immediately be teased out.

Working left to right in **Error! Reference source not found.**, the project specific community is the first difference. In this case, the project specific community is not identical to the wider F/OSS community served by F/OSS collaborative resources on the Internet. In the case of DoDSF, it is likely and expected that DoDSF will be gated in some manner, thus losing the 'F/O' as in F/OSS. The reality is that there will be classified software that the DoD hopes and expects to be reused and to evolve in a collaborative sense. Therefore, the openness assumed and intended for DoDSF will be as open as it can be for those in the gated community. This is not unprecedented; over the last decade, many private corporations—also wanting to reap the benefits of open and collaborative software development—have adapted F/OSS ideals. Such initiatives have been labeled using the terms corporate source (Dinkelacker, & Garg, 2001), progressive open source (Melian, 2007), and inner corporate source (Wesselius, 2006).

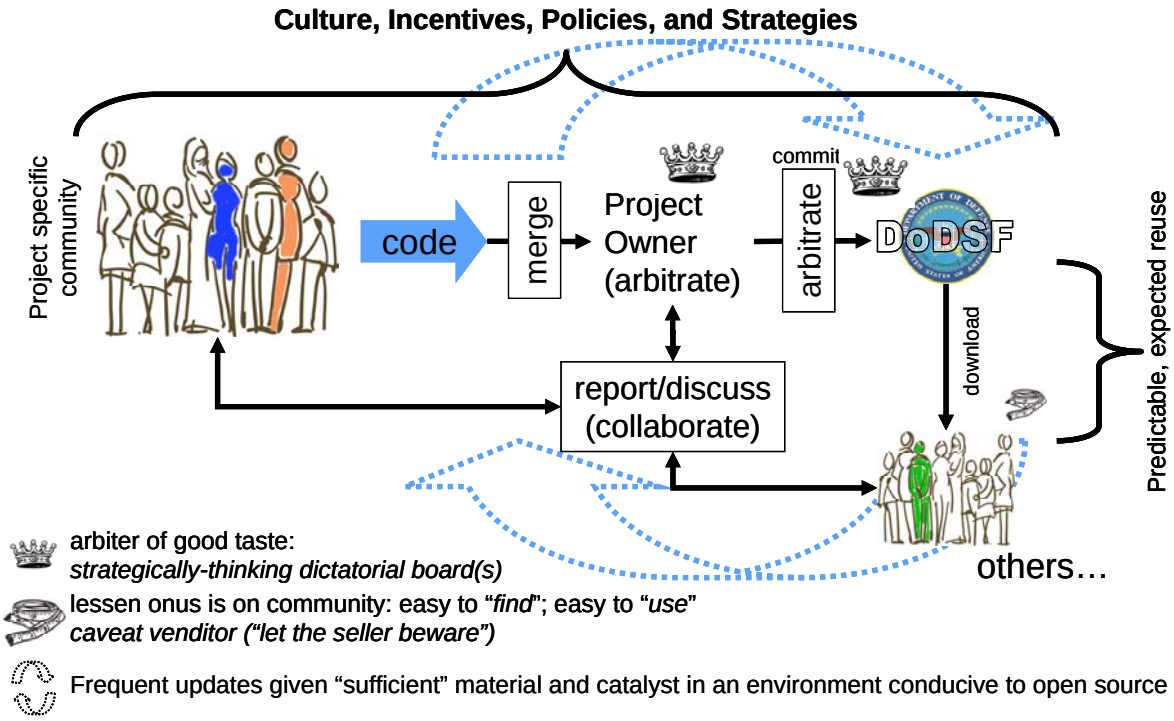

Figure 2. Abstract View of a DoDSF Project's Operation

The other difference in this community is its mix (as denoted by the shading of some of the characters in **Error! Reference source not found.**). Some from the community will likely be employees of private companies under contract to the DoD and under the oversight of a government program office—it is not assumed that these are the same private companies, contracts or government offices; it is only assumed that they share common needs and concerns. This too, is not unprecedented. In the F/OSS community, an

[2] This is not intended to be narrow, as we recognize that post deployment maintenance and long-term support would also have to benefit from open, collaborative and continuous software development. The description here is suitable for our discussion.

increasing number of private companies allocate resources to F/OSS projects and some companies even sponsor F/OSS projects, for example, MySQL, IBM for Eclipse, and Sun Microsystems for OpenOffice.org.

Moving further to the right in **Error! Reference source not found.**, the next significant difference is the introduction of an additional commit and arbitration step and a second crown. This abstraction is added to our DoDSF as a means to rectify weaknesses in the SourceForge.net abstraction discussed earlier regarding *caveat emptor* and the burden that is placed on the larger community having to assess a project artifact's degree of fit. As in F/OSS projects, it is expected that projects will continue to have "project owner(s)" that arbitrate (and delegate) ultimate control over what is committed into the project's code (or artifact) base, what software features are added or removed (over time), and the priorities upon which work progresses.

What is different with the introduction of the additional step is that these project owners are not the sole arbitrator as to what (specifically) from the project's codebase is actually committed to DoDSF. This additional arbitration step is needed to ensure that which is being submitted to DoDSF is consistent with the domain- or application-specific nature reflected onto DoDSF—in other words, the project's artifact is consistent with the architecture and variation mechanisms expected and needed for effective reuse of artifacts contained within DoDSF (Bachmann & Clements, 2005). How and who conducts that additional arbitration certainly would need to be addressed. Some software reuse repositories discussed earlier, specifically STARS SCAI and CARDS, used domain engineering approaches (i.e., domain managers) reflective of software product lines (i.e., product line manager) to oversee such consistency (USAF, 1996; Clements & Northrop, 2001). This, in effect, would empower the administrators or arbitrators (the second crown) of DoDSF with a role in quality arbitration not seen in SourceForge.net and reminiscent of earlier software reuse repositories, thereby affording the opportunity for a software product line approach.[3]

Given this additional step, the intent would be to reduce the real effort expended by others who find and assess artifacts downloaded from DoDSF for fitness for use and to increase the likelihood that those artifacts can be reused within their context (denoted by the smaller size of the measuring tape in **Error! Reference source not found.**). This represents a fundamental shift from the model in the F/OSS community of *caveat emptor* with the onus on the "buyer" to *caveat venditor*, or "let the seller beware," as the onus would shift to the product line managers to ensure that the artifacts committed to DoDSF are fit for (re-)use.

Continuing on the journey around **Error! Reference source not found.**, the next visual clue introduced is that in the lower right, depicting the group separate from the project specific group. This group is the same as that served in the F/OSS abstraction discussed earlier—a group that has come to DoDSF to find and reuse artifacts suitable for their context. However, this group has the foreknowledge that artifacts within DoDSF have been developed following product line practices. That would mean that DoDSF could have domain- and/or application-specific classifications, taxonomies, and software architectures that are meaningful to the DoD community and commonality across similar projects.

[3] Additional opportunities for collaboration are possible with the "project owners," including the supplier, users, and others in the DoD community with the arbitrators working this additional step.

Like **Error! Reference source not found.**, **Error! Reference source not found.** also includes cyclic, thick arrows to represent, in this case, a need for frequent updates to artifacts contained within DoDSF. Like the F/OSS community, the DoD community should also be continuous in its endeavor to improve the quality of its software through open and collaborative development. And, like its F/OSS counterpart, updates of artifacts to DoDSF should not be bound exclusively by fixed or planned milestones, as traditionally thought in contracted software acquisition. Rather, here, updates are driven by the DoD community.

Without this cyclic churning, for example, a project artifact is only submitted to DoDSF at or near the "completion" of a project; there then is no opportunity for DoD community feedback and participation in the open and collaborative process that is expected to improve quality or spark innovation. Inclusion of this cyclic churning is a significant break from the software development delivery pattern seen in the traditional DoD software acquisition lifecycle. To summarize key points taken from this DoDSF view:

- Like SourceForge.net, DoDSF would be a resource for IT-related services housing artifacts from DoD projects supporting open and collaborative development.
- Although the "project owner" has purview over the DoD project itself, the artifacts that are committed to DoDSF are arbitrated in a manner that is consistent with a product line approach.
- The DoD community here is a gated community similar to the F/OSS collaborative model adapted by private companies.
- The mantra of "release early, release often," indicative of F/OSS, is necessary to stimulate collaboration and spark innovation, as it does in the F/OSS community.

Throughout this discussion of DoDSF, it was assumed that the DoD community was motivated to behave in a manner that was consistent with the behavior often exhibited by the F/OSS community. We now turn our attention to this assumption.

Incentivizing a Culture of Collaboration, Innovation and Reuse

There is one final visual in **Error! Reference source not found.** to be discussed, that is the overarching "umbrella" of culture, incentives, policies, and strategies that must exist to engender the DoD community to behave in a manner that is indicative of openness and collaboration. The intent of this "umbrella" is to achieve the goals of reuse, quality and innovation coveted of the F/OSS community. Returning again to the OTD Roadmap, which recognized that their Roadmap "entails a parallel shift in acquisition methodologies and corporate attitude to facilitate discovery and re-use of software code across DoD." The Roadmap goes on to explain that today's acquisition model treats "DoD-developed software code as a physical good, DoD is limiting and restricting the ability of the market to compete for the provision of new and innovative solutions and capabilities." So any reformulation of today's acquisition model will fundamentally have to change the laws, policies and even thinking of the software code, not so much as a product, but more as means to mission capabilities and perhaps services. This is understandably a daunting task (white paper or not).

F/OSS Collaboration, Innovation and Reuse

Raymond's comprehensive insight into the motivation of the F/OSS community is foundational (Raymond, 2001). For some, necessity is the only impetus—a simple need for something. And, fortunately, many in the F/OSS community have the ability to fulfill that need through coding. And when their ability is outstripped by the realities of the problem, they create an F/OSS project and hope that others having the skills join (the birth of a project community). Such people that lend their helping hands often do with the greatest of intentions perhaps motivated by the same need or simply just feel the need to do some technically interesting work (i.e., "scratch an itch" in **Error! Reference source not found.**).

Sometimes that "need" can already be satisfied by product offerings from the commercial marketplace (i.e., the Cathedral) but the desire is to make a better alternative to that offering, one that is free and open to all. Many F/OSS projects started this way.

Figure 3. Culture of Collaboration in the F/OSS Community

As touched upon briefly above in Section 3, there is precedence for business models based on F/OSS projects. Many new projects have come and are coming into existence through software contributions *en masse* (e.g., Netscape's Mozilla, Sun's Java, IBM's Eclipse, MySQL) as business opportunities appear from ancillary services through the contribution of these codebases and through their use. However, this in and of itself is not an answer, but it certainly presents evidence to the behavior that is desirable in the DoD community. The Ultra-Large-Scale Systems (ULS) study called for research in Social and Economic Foundations for Non-Competitive Social Collaboration as inspired, in part, by the F/OSS movement; "as pure self-interest is supplanted by altruistic motivations and the desire to be perceived as productive and intelligent" while at the same time recognizing the need for incentive structures encouraging the community to cooperate (Feiler et al., 2006).

It is also important to recognize those that are motivated to voluntarily offer their time and contribute to F/OSS projects. Some of the motivations just discussed apply to these individuals as well (i.e., altruism, itching, etc.), but further extend to the meritocratic—that is to (socially and in governance)—rise in the community to which they serve. Further, some see F/OSS projects as venues to show off their prowess, to develop skills that make them more employable, or to network with others (a social phenomenon). And practically, others need (not just want) to see that their modifications, enhancements, and features find there way back into the mainstream product. Otherwise, if the F/OSS community does not accept

such changes, the only recourse is to reincorporate those changes into all future versions (i.e., rework) (Hissam & Weinstock, 2001).

Reasoning about DoDSF (Section 3) based on resources like SourceForge.net show DoDSF must differ if there is to be effective reuse for the DoD. For one, a DoD project is not likely to be incorporated in its entirety within some other DoD project. The projects are simply too big. However, there are certainly subsystems or modules of those overall projects that lend themselves to the DoDSF model. An example might be a subsystem that develops a common operational picture from a series of incoming tracks. To be able to reuse such a subsystem will require commonality at many levels, including mission needs, requirements, software architecture, design, data- and function-interdependencies, and other software artifacts.

Practically all of the Linux distributions (Debian, Fedora, Ubuntu, etc.) reuse the Linux kernel (www.kernel.org), which itself (Linux) has been ported to a wide variety of hardware architectures. In those distributions, other F/OSS applications are included (a list which is simply too long to even begin to enumerate). At the same time, like the Linux distributions, there are other POSIX-based distributions that are Linux-free, for example, Apple's Mac OS X, which is based on the Berkeley Software Distribution (BSD) of The Open Group's UNIX. And those same applications available to the Linux distributions are mostly available to Mac OS X. For the F/OSS community; the reasons for this are obvious: the underlying operating system, its architecture, interfaces (both for applications and device drivers), and interdependencies are openly specified, architected and, when necessary, debated. This leads to a shared understanding and context.

Baldwin and Clark (2006) argued that the architecture of F/OSS projects is a critical factor of the open and collaborative software development process in that it is the modularity of those architectures and the option values stemming from such modular architectures that contribute to collaboration and innovation. They noted that codebases that are more modular have more option value, thus attracting volunteers. That is the more modular and option rich, the more active and larger the innovator community is likely to be. Furthermore, it is these innovators that are incentivized to form voluntary, collective groups for the purpose of sharing and improving ideas. This, in and of itself, increases the likelihood of future variations and experimentation. Finally, the ULS report identified modularity as key to managing the complexity of software and to producing software systems amenable to change and to *concurrent* development—something that is clearly indicative of F/OSS collaborative development.

Looking again to some of the F/OSS "poster children," specifically Linux, Apache, and now Firefox (direct descendent of Netscape), those projects did not start out with wonderfully modular architectures. They only became modular after the complexity of features, project management, distributed development became too overwhelming and had to adapt. Chastek, McGregor, and Northrop (2007) identified Eclipse's plug-in (modular) architecture as one of the project's most valuable core assets, providing for multiple forms of variation including extension points of various types and (in the authors' opinion) learning from the lessons from past F/OSS projects.

To summarize key points taken from this F/OSS view:

- Some of the incentives that motivate individuals, groups, and companies to participate and collaborate in the F/OSS community can be explained, but more study is warranted.

- Some private companies have moved from treating software source code as a physical good and have found market opportunity in services from the use of the software.
- Modularity of an architecture not only promotes reuse, but is a key factor in spurring innovation in collaborative communities.
- Like F/OSS projects, software emanating from DoD projects will have to have architectures and interfaces that promote modularity and option value.

DoDSF Collaboration, Innovation and Reuse

Taking the key points from the previous sections, the "big money" question is how do these map into the gated DoD community that was established back in Section 3 (recall civilian and military personnel, along with employees of private companies under contract to the DoD and under the oversight of a government program office having common needs and concerns)? Furthermore, what needs to be done to change acquisition policy and strategy and to establish the incentives that will enable a culture and behavior similar to that seen in the F/OSS community?

As daunting as these questions may be, we humbly offer a few suggestions.

Recognize Product Line Practices are Not Free

Creating modularized subsystems and components that are consistent with the architecture and variability expected and needed for effective reuse will cost development dollars with payoff that may not be realized until the reuse of the component can be amortized. **Strategically, this should be expected and not avoided.** Furthermore, and before new components are created (or existing components are refactored), resources will have to be expended to identify product-line-wide architectures that are suitable for DoDSF and against which project artifacts are assessed before commitment to DoDSF. Such activities will likely require planning and development that are beyond any one project, yet are necessary for the projects themselves. Such planning includes mission objectives, product strategies, requirements analysis, architecture and design modifications, extra documentation, and packaging. Incentivizing the program managers that oversee these projects would require some combination of providing extra funding and making performance evaluation dependant on contributions to DoDSF.

Incentivize the "Churn"

If effort is to be expended to create a product-line-wide architecture for the DoDSF, and individuals across the DoD-wide enterprise are empowered as product line managers, the DoDSF has to be more than a "field of dreams" followed by the often cursing mantra "If you build it, they will come." Recognize that reuse is not free and that reuse does not come easily or by happenstance (Tracz, 1995). If the desired behavior of the DoD community is to use the DoDSF for finding project artifacts, then those artifacts have to be meaningful, relevant, and, by reputation, sound. Recall, the desire is to unburden the "buyer" from assessing the component's degree of fit—as expected in software product lines. By reducing this burden as a significant barrier to reuse, incentives may be necessary to bootstrap or kick start reciprocating contributions, feedback, improvements, and otherwise collaborative behaviors—but observations from the F/OSS community would lead to the belief that such incentives would not be necessary. But this is not entirely clear in the gated

DoD community. Talented, willing and able civilian and military personnel may be more likely to behave in this manner. Employees of private companies—while on contract—might also behave in this manner. Again, there is precedent in the F/OSS community for private companies to commit resources to F/OSS projects. Following this model, perhaps there are incentives for contracting companies that are successful in getting subsystems and components into DoDSF—that being negotiated service contracts, thereby allowing for continued involvement servicing the DoD community.

There are good reasons (perhaps un-incentivized) that a new DoD project would prefer to see bidders propose using proven artifacts from DoDSF. Such includes less risk to the project—a subsystem taken from DoDSF is already a known quantity, and lower development costs allowing valuable program dollars to be used elsewhere in the program. A possible disincentive (or opportunity, perspective is everything) is that it may be viewed by Congress that the project should be built for less money because it uses a subsystem(s) in DoDSF; the program office may be given less money to get the job done, which may be viewed as a negative outcome by some.

A supplier bidding on a project really has only two incentives to use an artifact contained in DoDSF. If the program office has indicated that the use of such artifacts will be a determining factor in a successful proposal, then there is a strong incentive to do so. In the absence of such a requirement, the supplier may be incentivized to reuse an artifact to enable it to be the lowest bidder.

Incentivize Software as a Non-Rivalrous Good

Treating source code as if it were a physical good is a mentality that inhibits collaboration. Rivalry should be encouraged between competing subsystems or components for the same role in a produce-line-wide architecture (i.e., let the stronger or better prevail). But the source code itself should serve as the source of inspiration, innovation and improvements for that “better” subsystem—rather than the opaque enigma requiring resources to be expended to re-engineer from scratch (or worse, reverse-engineer because the source code is long forgotten and lost).

Last Thoughts

Governance

Reminiscent of reuse repositories discussed in Section 2, great care has to be given in governance of DoDSF. The DoD must have a vested interest in seeing that the artifacts in DoDSF can be reused in subsequent projects. It has invested in them and would like to see a payback in terms of reduced development time, risk, and cost in the future. Thus, there is an upfront quality requirement for items to be placed into DoDSF. For SourceForge.net, the evaluation is ultimately done by the F/OSS community (using or not using) the project. For DoDSF there is presumably a contractual requirement regarding the subsystem. Someone has to evaluate the subsystem and its suitability for reuse, which needs to be a part of the original development contract. Otherwise there is every incentive for the supplier to place something into DoDSF that is ultimately unusable by anyone other than the original supplier.

Who does this evaluation? In the body of this paper, we placed the onus on the “seller” (*caveat venditor*), which, in this case, was tagged as the product line manager or the “second crown.” In reality, that role will come down to real people in the DoD community.

Determining just who exactly those individuals are is beyond the scope of this white paper, but it is certainly something that will have to be decided.

Security

In this white paper, we acknowledge that classification of project artifacts in DoDSF is a reality. This presents a challenge for DoDSF. If an artifact is from a top-secret project, then it may be difficult to declassify it for contribution to a DoDSF that does not respect security issues. But allowing DoDSF to embrace a multi-level security model raises concerns. Here's one example. Is a top secret project able to use an artifact classified at a lower level? If so, how does it trust it? If it makes modifications (even a bug fix) what happens to the security classification of the artifact when the modification is given back to DoDSF? Does this result in a security-level fork? There are many such questions that could be raised, but a further discussion of this is beyond the scope of this paper.

Summary

The number of references that were used in the preparation of this white paper was far more than any of the authors expected. This simply illustrates, in our opinion, the tip of a very large iceberg on the topic of reuse and F/OSS openness and collaboration coming from various disciplines.

Perhaps the most relevant reference that we came across for this paper was the Open Technology Development Roadmap Plan (Herz et al., 2006). Those interested in following up on some of the discussion covered in this paper should consider getting the latest progress on the actions called for within that Roadmap Plan. That plan called for very specific actions with respect to changing the traditional acquisition lifecycle. Most interesting was the recommendation: "Evaluate the potential use of the Defense Acquisition Challenge (DAC) program to demonstrate Open Technology alternatives to projects or programs that have implementation issues; e.g., make application of open source based products or development methodologies a specific interest item for DAC."

On the topic of product lines, it is worth noting that there are case studies that show how product line approaches can be effective and successful in industry and government ventures (USAF, 1996; Clements & Northrop, 2001; Jensen, 2007; Mebane & Ohta, 2007). Furthermore, there are efforts and thinking happening now to merge F/OSS models with software product lines (Chastek et al., 2007) and (van Gurp, Prehofer & Bosch, 2010) along with three international workshops on Open Source Software and Product Lines (specifically OSSPL 2006, OSSPL 2007, and OSSPL 2008).

F/OSS works today because of the culture, environment, and motivation touched upon in this white paper. It is important to note that this F/OSS culture was not planned at all, but is founded by a loose set of principles and rules (some of which are formalized through F/OSS licenses) that guide the behavior to achieve freely available, lightly controlled software developed in a collaborative manner. This behavior is informed by centuries of human populations and communities creating new knowledge and building off each other's work.

The question the readers should ask themselves (and we would not have done our job if you didn't ask yourself) is what would such principles and rules look like in a gated DoD community, a community itself informed by approximately 200 years of contracting,

procurement and competition? Additionally, what is needed to foster the behavior the DoD wants to engender? What can the DoD control and what control must the DoD relinquish?

Acknowledgements

We would like to thank Gary Chastek, Terry Dailey, Bob Gobeille, Guy Martin, Catherina Melian, Linda Northrop, Robert Vietmeyer, and Kurt Wallnau for their thoughtful review and suggestions and with a special thanks to Nickolas Guertin whose curiosity, energy, and interest in the topic inspired this paper.

References

Bachmann, F., & Clements, P. (2005). *Variability in software product lines* (CMU/SEI-2005-TR-012). Pittsburgh, PA: Carnegie Mellon Software Engineering Institute.

Baldwin, C.Y., & Clark, K.B. (2006). The architecture of participation: Does code architecture mitigate free riding in the open source development model? *Management Science, INFORMS*, *52*(7), 1116-1127.

Brachman, R.J., & Schmolze, R.J. (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science*, *9*(2), 171-216.

Chastek, G.J., McGregor, J.D., & Northrop, L.M. (2007). *Observations from viewing eclipse as a product line*. Second International Workshop on Open Source Software and Product Lines 2007 (OSSPL07). Limerick, Ireland.

Clements, P., & Northrop, L. (2001). *Software product lines: Practices and patterns*. Boston, MA: Addison Wesley Professional.

Department of the United States Air Force (USAF). (1996). *Guidelines for successful acquisition and management of software intensive systems* (Vers. 2.0). Software Technology Support Center.

Dinkelacker, J., & Garg, P.K. (2001). *Corporate source: Applying open source concepts to a corporate environment*. First International Workshop on Open Source Software Engineering. The 23rd International Conference on Software Engineering (ICSE). Toronto, Canada.

Feiler, P., Gabriel , R., Goodenough, J., Linger, R., Longstaff, T., Kazman, R., Klein, M., Northrop, L., Schmidt, D., Sullivan., & Wallnau, K. (2006). *Ultra-large-scale systems: The software challenge of the future*. Retrieved from http://www.sei.cmu.edu/library/assets/ULS_Book20062.pdf

Feller, J., & Fitzgerald, B. (2001). *Understanding open source software development*. Boston, MA: Addison Wesley Professional.

Frakes, W.B., & Nejmeh, B.A. (1987). Software reuse through information retrieval. *SIGIR Forum*, *21*(1-2), 30–36.

Gardner, D. (2005, August 17). SHARE, IBM user group, to celebrate 50th anniversary. *InformationWeek*. Retrieved from http://www.informationweek.com/news/showArticle.jhtml?articleID=169400167

Goldman, R., & Gabriel, R.P. (2005). *Innovation happens elsewhere*. San Francisco, CA: Morgan Kaufmann.

Granoff, M. (2002). *The story of SIMTEL20*. FARNET Stories Project, Coalition for Networked Information, Interop, Inc., and the National Science Foundation. Retrieved March 2009, from http://www.cni.org/docs/farnet/story149.NM.html

Herz, J.C., Lucas, M., & Scott, J. (2006, April). *Open technology development roadmap plan*. Prepared for Ms. Sue Payton, Deputy Under Secretary of Defense, Advanced

Systems & Concepts. (Vers. 3.1). Retrieved from http://www.acq.osd.mil/jctd/articles/OTDRoadmapFinal.pdf

Hissam, S., & Weinstock, C. (2001). *Open source software: The other commercial software*. First International Workshop on Open Source Software Engineering; 23rd International Conference on Software Engineering (ICSE). Toronto, Canada. Retrieved from http://opensource.ucc.ie/icse2001/hissamweinstock.pdf

Howe, W. (2009). *A brief history of the Internet*. Walt Howe's Internet Learning Center. Retrieved March 2009, from http://www.walthowe.com/navnet/history.html

Jackson, J. (2008, October 8). Pentagon: Open source good to go. *Government Computing News*. Retrieved from http://gcn.com/articles/2008/10/08/pentagon-open-source-good-to-go.aspx

Jensen, P. (2007). *Experiences with product line development of multi-discipline analysis software at Overwatch Textron Systems*. 11th International Software Product Line Conference (SPLC 2007). Kyoto, Japan.

Kempe Software Capital Enterprises. (1998). *ASSET: The asset source for software engineering technology*. Ada Home. Retrieved March 2009, from http://www.adahome.com/Network/Repositories.html

Mebane H., & Ohta, J.T. (2007). *Dynamic complexity and the Owen Firmware Product Line Program*. 11th International Software Product Line Conference (SPLC 2007). Kyoto, Japan.

Melahn, W. (1956). A description of a cooperative venture in the production of an automatic coding system. *Journal of the ACM (JACM)*, *3*(4), 266-271.

Melian, C. (2007). *Progressive open source: The construction of a development project at Hewlett-Packard*. Unpublished doctoral dissertation, Stockholm School of Economics, Sweden.

Raymond, E.S. (2001). *The cathedral & the bazaar: Musings on linux and open source by an accidental revolutionary*. Sebastopol, CA: O'Reilly Media.

Schaefer, T.M. (2005, January). [Letter to the Editor]. *Crosstalk, The Journal of Defense Software Engineering*, 18(1). Hill AFB, Utah. Retrieved from http://www.stsc.hill.af.mil/CrossTalk/2005/01/0501LetterToEditor.html

SourceForge, Inc. (2009a). *Terms of use*. Retrieved March 2009, from http://alexandria.wiki.sourceforge.net/Terms+of+Use

SourceForge, Inc. (2009b). *What is SourceForge.net?* Retrieved March 2009, from http://alexandria.wiki.sourceforge.net/What +is+SourceForge.net%3F

Stenbit, J.P. (2003, May 28). *Open source software (OSS) and the Department of Defense (DoD)*. Memorandum from the United States Department of Defense. Retrieved from http://cio-nii.defense.gov/docs/OpenSourceInDoD.pdf

Tracz, W. (1995). *Confessions of a used program salesman: Institutionalizing software reuse*. Boston, MA: Addison Wesley Longman.

van Gurp, J., Prehofer, C., & Bosch, J. (2010). Comparing practices for reuse in integration-oriented software product lines and large open source software projects. *Software: Practice and Experience*, *40*(4), 285-312.

Wallnau, K.C. (1992). An introduction to CARDS. *Crosstalk, The Journal of Defense Software Engineering*, (32). Hill AFB, Utah.

Wesselius, J. (2006). The bazaar inside the cathedral: Business models for internal markets. *IEEE Software*, *25*(3), 60-66.

THIS PAGE INTENTIONALLY LEFT BLANK

2003 - 2010 Sponsored Research Topics

Acquisition Management

- Acquiring Combat Capability via Public-Private Partnerships (PPPs)
- BCA: Contractor vs. Organic Growth
- Defense Industry Consolidation
- EU-US Defense Industrial Relationships
- Knowledge Value Added (KVA) + Real Options (RO) Applied to Shipyard Planning Processes
- Managing the Services Supply Chain
- MOSA Contracting Implications
- Portfolio Optimization via KVA + RO
- Private Military Sector
- Software Requirements for OA
- Spiral Development
- Strategy for Defense Acquisition Research
- The Software, Hardware Asset Reuse Enterprise (SHARE) repository

Contract Management

- Commodity Sourcing Strategies
- Contracting Government Procurement Functions
- Contractors in 21st-century Combat Zone
- Joint Contingency Contracting
- Model for Optimizing Contingency Contracting, Planning and Execution
- Navy Contract Writing Guide
- Past Performance in Source Selection
- Strategic Contingency Contracting
- Transforming DoD Contract Closeout
- USAF Energy Savings Performance Contracts
- USAF IT Commodity Council
- USMC Contingency Contracting

Financial Management

- Acquisitions via Leasing: MPS case
- Budget Scoring
- Budgeting for Capabilities-based Planning

- Capital Budgeting for the DoD
- Energy Saving Contracts/DoD Mobile Assets
- Financing DoD Budget via PPPs
- Lessons from Private Sector Capital Budgeting for DoD Acquisition Budgeting Reform
- PPPs and Government Financing
- ROI of Information Warfare Systems
- Special Termination Liability in MDAPs
- Strategic Sourcing
- Transaction Cost Economics (TCE) to Improve Cost Estimates

Human Resources

- Indefinite Reenlistment
- Individual Augmentation
- Learning Management Systems
- Moral Conduct Waivers and First-tem Attrition
- Retention
- The Navy's Selective Reenlistment Bonus (SRB) Management System
- Tuition Assistance

Logistics Management

- Analysis of LAV Depot Maintenance
- Army LOG MOD
- ASDS Product Support Analysis
- Cold-chain Logistics
- Contractors Supporting Military Operations
- Diffusion/Variability on Vendor Performance Evaluation
- Evolutionary Acquisition
- Lean Six Sigma to Reduce Costs and Improve Readiness
- Naval Aviation Maintenance and Process Improvement (2)
- Optimizing CIWS Lifecycle Support (LCS)
- Outsourcing the Pearl Harbor MK-48 Intermediate Maintenance Activity
- Pallet Management System
- PBL (4)
- Privatization-NOSL/NAWCI
- RFID (6)

- Risk Analysis for Performance-based Logistics
- R-TOC AEGIS Microwave Power Tubes
- Sense-and-Respond Logistics Network
- Strategic Sourcing

Program Management

- Building Collaborative Capacity
- Business Process Reengineering (BPR) for LCS Mission Module Acquisition
- Collaborative IT Tools Leveraging Competence
- Contractor vs. Organic Support
- Knowledge, Responsibilities and Decision Rights in MDAPs
- KVA Applied to AEGIS and SSDS
- Managing the Service Supply Chain
- Measuring Uncertainty in Earned Value
- Organizational Modeling and Simulation
- Public-Private Partnership
- Terminating Your Own Program
- Utilizing Collaborative and Three-dimensional Imaging Technology

A complete listing and electronic copies of published research are available on our website: www.acquisitionresearch.org

THIS PAGE INTENTIONALLY LEFT BLANK

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL
555 DYER ROAD, INGERSOLL HALL
MONTEREY, CALIFORNIA 93943

www.acquisitionresearch.org