2018-06

SCALED APPROACH TO OPEN SOURCING DEPARTMENT OF THE NAVY PRODUCED SOFTWARE

Garcia, Julian L.; Holloway, Donovan Jr.

Monterey, CA; Naval Postgraduate School

https://hdl.handle.net/10945/60404

THESIS

SCALED APPROACH TO OPEN SOURCING DEPARTMENT OF THE NAVY PRODUCED SOFTWARE

by

Julian L. Garcia and Donovan Holloway Jr.

September 2018

Thesis Advisor:	Karen L. Burke
Second Reader:	Glenn R. Cook

REPORT DOCUMENTATION PAGE				*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.				
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2018	**3. REPORT TYPE AND DATES COVERED** Master's thesis		
4. TITLE AND SUBTITLE SCALED APPROACH TO OPEN SOURCING DEPARTMENT OF THE NAVY PRODUCED SOFTWARE				**5. FUNDING NUMBERS** W7A16
6. AUTHOR(S) Julian L. Garcia and Donovan Holloway Jr.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000				**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) DoN Secretariat Historian, Arlington, VA 22202				**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.				**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

The Department of Defense (DoD) must continue to develop, sustain, and update its software-based capabilities. For the Department of the Navy (DoN), the life cycle costs of software continue to grow; over time, developing code will not be cost effective. An alternative to developing code is to further integrate open source software (OSS) into DoN programs. OSS is software that grants users the ability to view, use, and change the software source code. The use of OSS has been extensively researched, as addressed in the MITRE Corporation's study on free and open source software (FOSS) in the DoD, completed in 2003. Despite favorable reports and published DoD policy, and the widespread successful use of OSS in current software, program managers are reluctant to fully integrate OSS into the DoN due to concerns with legal requirements, cybersecurity, total expenses, and the ability to implement and control OSS on classified systems while adhering to security regulations. This study utilized a quantitative, scaled approach to determine the risks and benefits to open sourcing for all DoN software. Several OSS case studies were examined. This research concluded that while OSS has been tested and proven cost-effective in certain areas of the DoN, it may not be the most efficient solution for all DoN projects. Therefore, the DoN should consider further implementation of OSS in security, software development, infrastructure support, and for program lifecycle cost reductions.

14. SUBJECT TERMS open source software, open source code, free and open source			**15. NUMBER OF PAGES** 101
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

SCALED APPROACH TO OPEN SOURCING DEPARTMENT OF THE NAVY PRODUCED SOFTWARE

Donovan Holloway Jr.
Captain, United States Marine Corps
BSBA, Auburn University, 2012

Julian L. Garcia
Captain, United States Marine Corps
BBA, Prairie View A & M University, 2011

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 2018

Approved by: Karen L. Burke
Advisor

Glenn R. Cook
Second Reader

Dan C. Boger
Chair, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Department of Defense (DoD) must continue to develop, sustain, and update its software-based capabilities. For the Department of the Navy (DoN), the life cycle costs of software continue to grow; over time, developing code will not be cost effective. An alternative to developing code is to further integrate open source software (OSS) into DoN programs. OSS is software that grants users the ability to view, use, and change the software source code. The use of OSS has been extensively researched, as addressed in the MITRE Corporation’s study on free and open source software (FOSS) in the DoD, completed in 2003. Despite favorable reports and published DoD policy, and the widespread successful use of OSS in current software, program managers are reluctant to fully integrate OSS into the DoN due to concerns with legal requirements, cybersecurity, total expenses, and the ability to implement and control OSS on classified systems while adhering to security regulations. This study utilized a quantitative, scaled approach to determine the risks and benefits to open sourcing for all DoN software. Several OSS case studies were examined. This research concluded that while OSS has been tested and proven cost-effective in certain areas of the DoN, it may not be the most efficient solution for all DoN projects. Therefore, the DoN should consider further implementation of OSS in security, software development, infrastructure support, and for program lifecycle cost reductions.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ALM	Application Life Cycle Management
API	application programming interface
ASR	authorized strength report
CGMCS	Coast Guard Machinery Control System
CECOM	Communication Electronics Command
CIO	Chief Information Officer
CMC	Commandant of the Marine Corps
CNSSP	Committee on National Security Systems Policy
COTS	commercial-off-the-self
CSIS	Center for Strategic and International Studies
CSS	closed source software
DAU	Defense Acquisitions University
DevOps	development and operations
DISA	Defense Information Systems Agency
DoD	Department of Defense
DoN	Department of the Navy
DTS	Defense Travel System
FCC	Fleet Cyber Command
FFRDC	federal funded research and development centers
FINTECH	financial technology
FOSS	free open source software
GAO	Government Accountability Office
GOTS	government-off-the-shelf
GPL	general public license
GPS	global positioning system
GTC	government travel card
HP	Hewlett-Packard
HTTP	hypertext transfer protocol
ISR	intelligence, surveillance, and reconnaissance
IDC	International Data Corporation

IT	information technology
LIMUX	Linux in Munich
MARADMIN	Marine Corps administrative message
MCS	Machinery Control system
MOL	marine online
MRRS	medical readiness reporting system
NMCI	Navy and Marine Corps Intranet
NIST	National Institute of Standards and Technology
NIPR	non-classified Internet protocol router
OAI	Open Application Initiative
OCCFLD	operations occupational field
OIC	officer-in-charge
OMB	Office of Management and Budget
OSI	Open Source Initiative
OSS	Open Source Software
OTD	open technology development
SARA	Security Auditor’s Research Assistant
SIPR	secure Internet protocol router
TCO	Total Cost of Ownership
URL	uniform resource locator
USCYBERCOM	United States Cyberspace Command
VISTA	veteran’s health information systems and technology architecture
VPAT	voluntary product accessibility template

ACKNOWLEDGMENTS

We would like to start by thanking our thesis advisor, Professor Karen Burke, for her guidance, patience, support, and the time that she devoted to this thesis. We were incredibly grateful to have such a caring and no-nonsense leader as a mentor throughout our Naval Postgraduate School experience.

We would also like to thank our second reader, Professor Glenn Cook. His knowledge, patience, and guidance were paramount during the thesis process and time spent learning and growing at NPS. We are thankful for both our advisor and second reader and want to thank them in advance for the advice we anticipate in our future endeavors.

Finally, we are especially grateful for the love and support from our families. Words cannot express how important our family support has been during this process.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The life cycle costs of software continue to grow for the Department of the Navy (DoN), and the fear is these costs will become unsustainable. The use of open source software (OSS) has been an interest of many in the community since the 1990s (Endsley, 2018). The use of OSS in the Department of Defense (DoD) has been studied for over a decade; yet, the DoD and the DoN continue developing code at an incredible expense. The DoD addresses the use of OSS, maintains policies for its use, and provides cybersecurity controls to be applied. However, there seem to be perceptions by some managers in the DoD that use of open source code is not a feasible option; consequently, code development continues. Given the potential cost benefits of using OSS, it is necessary to explore the feasibility of using OSS in the DoN.

The research reviewed the current reports on open sourcing, which included: "OSS in Government, Challenges and Opportunities," dated August 2013; and "Open Technology Development (OTD): Lessons Learned & Practices for Military Software," dated May 2011. The research also explored the successes that have been discovered in the international software developer community and attempted to ascertain the reasons for their success. In addition to case studies on OSS, other topics such as OSS costs trends, vulnerabilities, and OSS key drivers were identified and discussed.

This research was applied to the DoN to determine the benefits and risks to more routine open-sourcing for DoN all software. With the knowledge of OSS trends and drivers, and the successful efforts of organizations around the world, this research was created to inform DoN leadership of potential courses of action for the DoN. Given potential limitations due to cybersecurity or legal requirements, a scaled approach proved to be more prudent, and was discussed as part of this research. Furthermore, the research reviewed current policies that specifically addressed the use of open sourcing, such as DoD memorandums and other published policy.

A. OBJECTIVES

The research is a quantitative study that relied on previously published literature, cost benefit analysis, previously collected data, software literature, successes of international software development, and an analysis of cost, risk, and benefits. The desired outcome of this research was to propose the feasibility of the use of OSS for DoN programs.

The research identified several factors that could provide recommended support for OSS in the DoN. Such factors included the following: the processes, criteria, and responsibilities for publishing publically-releasable software, current programs in the DoD that have incorporated OSS, cost benefits of OSS, and cybersecurity requirements and challenges of OSS.

This research was designed to determine the feasibility of OSS in applicable or manageable situations within the DoN. The recommendations of the research provided evidence that using a combination of open source and developed code, in a scaled approach, may be the best way to implement OSS for future projects. A cost benefit analysis considered the use of total open source components versus proprietary or closed-source components.

B. RESEARCH QUESTIONS

- What are the processes, criteria, and responsibilities for publishing publicly releasable software?
- What is the feasibility of using OSS for DoN programs, from unclassified through classified?
- Have there been any programs in the federal government, DoD, DoN that have incorporated OSS?
- Was there a discernible cost benefit for the program? How would/could OSS affect maintenance costs?

- What are the cybersecurity requirements or challenges for implementing OSS?
- If it were not possible to obtain all of the DoN’s requirements using OSS, would a scaled approach be cost effective?

C. THESIS DESIGN

Chapter II provides the reader with an overview of OSS and the current DoN policies regarding OSS. The chapter begins by establishing the definition of OSS and then continues by looking into studies conducted on the use of OSS. Additionally, the chapter examines differences between OSS, commercial-off-the-shelf software, and proprietary software. Finally, the chapter reviews OSS variables that must be considered when using OSS.

Chapter III continues the research with an examination of the benefits and vulnerabilities of OSS. The chapter presents a number of cost variables as well as intangible cost considerations. Here, the acquisition process is reviewed and case studies are used to demonstrate the costs, benefits, and vulnerabilities of using OSS. Lastly, the chapter concludes with a summary and advantages that were drawn from the current research.

Chapter IV begins with an overview of significant implementation steps needed to further integrate OSS into certain DoN produced software. In addition to the implementation process, this chapter explores the research findings regarding strategic goals, security concerns, and the Development and Operations (DevOPs) concept. The research recognizes DevOps as a concept that joins the developments and operations teams while working toward process efficiency and achieving strategic goals.

Finally, Chapter V briefly considers a holistic view of the research conducted throughout this thesis, addresses the research questions, and establishes recommendations for follow-on research.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

The use of OSS in the DoN allows the users to have the right to develop code and make changes as they see fit. The DoN program users and software engineers can use OSS to develop, study, and assemble software applications that will support the on-going needs of software updates in DoN programs. It is widely accepted that “Software that qualifies as free almost always also qualifies as open source, and vice versa, since both phrases derive from the same set of software user rights formulated in the late 1980s by Richard Stallman of the Free Software Foundation” (MITRE Corp, 2003, p. 2). In 1983, the GNU’s Not Unix (GNU) platform was created by Richard M. Stallman. The GNU was first designed in order to operate like UNIX, yet the software was free (Free Software Foundation, 2018). This was a new concept in the software world because other developers could use and manipulate the software without having to pay for it. Stallman proved to be an innovator and his initiative was one of the platforms that led to the development of OSS. Later, Stallman went on to create the Free Software Foundation of 1985 (Free Software Foundation, 2018). Today, GNU is largely associated with open source licenses. The free licenses of OSS are extensive and must be considered in the decision to use OSS. Deciding which or what combination of open source licenses to use can be complicated but must be done as a part of the open source selection process (Almeida, Murphy, Wilson, & Hoye, 2018).

The MITRE Corporation report defines free open-source software (FOSS) as, “software that gives users the right to run, copy, distribute, study, change, and improve it according to their needs, without them having to ask permission from or make additional payments to any external group or person” (MITRE Corp, 2003, p. 2). OSS allows users to view the source code, which in many cases, can expedite the process of writing code. “Source code” is software that works behind the scenes. Most computer users are unaware of how the source code operates but computer programmers can use this code to manipulate the way software—a “program” or an “application” works (Endsley, 2018). The code is sometimes referred to as the “blueprint” of a program. A programmer can read these blueprints and, like an architect, use them to create anything from a simple diagram of a digital tree house to a complex model of the Sydney Opera House, whatever the situation calls for. The point being that possession and the ability to manipulate code or blueprints opens up endless possibilities to engineers. According to Opensource.com, “Programmers who have access to a computer program’s source code can improve that program by adding features to it or fixing parts that do not always work correctly” (Endsley, 2018.). The traditional closed-source or proprietary option does not allow users to see or manipulate the source code, which can make it extremely difficult for an organization to adjust or manipulate the code to fit their needs. In the article, “What is Open Source?,” the term closed-source is described as follows: “Proprietary or Closed Source Software, is software that has source code that only the person, team, or organization who created it-and maintains exclusive control over it-can modify” (Endsley, 2018). This means that the DoN has to spend the resources of time, money, and manpower to write software from scratch, find a commercial-off-the-self option (COTS), or pay the original developer for modifications. Creating software takes time and experience. The DoN would have to be extremely flexible and efficient to keep up with exponential technological advances and cyber security. Alternatively, a COTS option may not meet all of the complex functionality needs and may require a high investment for modification and support.

Imagine using an available presentation template for a complex presentation vice creating a presentation from scratch. When a person undertakes a task, that person can take the time to gather all of the information, plan a presentation method, and then build a presentation with intricate detail and formulas. It would take a lot of time to perfect the presentation and make it effective. So much so, that creating a presentation in this method almost never happens. Typically, one usually simply opens Microsoft PowerPoint and selects a template; saving time on the basics of presentation development. The user would still have to tailor the presentation to meet specific needs but at least there is a completed presentation foundation. If the user does not have access to Microsoft PowerPoint, the user could use an open source presentation program such as LibreOffice Impress, if the user is in an environment that allows users to download and install software. That is not the case for DoD users.

In this situation, time and money and possibly cyber security policy are enormous barriers to production. What if a presentation was required, but there was not enough time to start from scratch or money to purchase the PowerPoint software? What if the software did not have the specific template that was needed and there was no way to manipulate the software to produce the desired result without signing an expensive extended contract for general support? This analogy roughly describes what and why OSS is beneficial. OSS is free, has various licenses to ensure ease of continued use, is versatile, tailorable, and is supported by developers worldwide (Free Software Foundation, 2018). Emily Rose, lead developer Evangelist at Salesforce, noted that having a tested starting point, with certain established OSS, will allow for faster, and more specific, software and technology development (Chrzanowska, 2017). These capabilities would assist the military in keeping up with the new challenges in the IT environment. The Microsoft PowerPoint analogy provides a good example of the importance and the theory of OSS, but the DoN must also explore how technology leaders view OSS.

In 2005, Microsoft founder Bill Gates referred to open source as “a new form of communism” (Asay, 2016, p. 1). At the time, Microsoft was thriving, boasting revenue increases of roughly $3 billion; bringing the yearly total to $39.79 billion (Gates & Ballmer, 2005). In a five-year span, from 2000 to 2005, the company’s revenues increased 73%, which equated to approximately $162 billion dollars in total revenue (Gates & Ballmer, 2005). With profits like these, one can plainly see how Gates would not be a proponent of free software. Today, Microsoft has openly embraced OSS and has even publicly announced its support for Linux (Asay, 2016). So, what changed? Why would a company such as Microsoft, that was making billions of dollars from its software and products, support free OSS? Perhaps Microsoft had to change its strategy because of the emergence of new competitors and technologies; or maybe the company managers were simply feeling generous and decided that using and producing OSS aligned with their morals. Whatever the reason, it is worth looking into open-source software, and why a company as successful as Microsoft now embraces OSS. In June 2018, Microsoft took a step further into the open source community by purchasing GitHub for $7.5 billion (Warren, 2018). Big technology companies and software developers have used GitHub as an open source code repository for years (Warren, 2018). With the purchase of GitHub and Microsoft's other projects such as "PowerShell, Visual Studio Code, and the Microsoft Edge Java script," which are now open source, it is plain to see that industry leaders are embracing OSS (Warren, 2018, p. 1). Currently, OSS is now intertwined in many aspects of the current technological environment and it is important to dive into the open source world and how it can be applied to the DoN.

A. OSS IN THE DOD

Wennergren argues that many leaders in the DoD do not understand OSS (Wennergren, 2009). Given the definition of OSS and the secret nature of many programs in the military, it seems logical that the military should steer clear of having anything, including software code, that is transparent to their adversaries. However, research has shown that the DoD already uses and has policy on OSS.

One commonly confusing part of DoD IT policy is found in the DoD Information Assurance (IA) policy. According to the DoD IA policy, Enclosure 4 [SI-7 (14)],

> Public Domain Software Controls Binary or machine executable public domain software products and other software products with limited or no warranty such as those commonly known as freeware or shareware are not used in DoD information systems unless they are necessary for mission accomplishment and there are no alternative IT solutions available. (Department of Defense Chief Information Officer, n.d., p. 1)

This statement refers to software in which "the Government does not have access to the original source code" and was clarified in a 2009 DoD Chief Information Officer (CIO) Memorandum (Department of Defense Chief Information Officer, n.d., p. 1). The clarifying statement explained that since the DoD would have access to the source code of OSS, then the previous control as set forth in the DoD IA policy does not forbid its use (Department of Defense Chief Information Officer, n.d.). As a result, OSS is approved for use in the DoD.

In an environment where security and IT is an imperative asset to any military, the DoD is in a transition to raise and build its technical capabilities. In the article "OSS and the Department of Defense," FitzGerald, Parziale, and Levin emphasize those capabilities and their dependence on software. The authors highlight that the current military advantage resides in their capabilities (FitzGerald, Parziale, & Levin, 2016). The DoD can expertly conduct intelligence, surveillance, and reconnaissance (ISR) missions, and employ precision munitions and targeting through the use of global positioning systems (GPS). These types of missions are largely dependent on IT capabilities, which means that the quality of the software is crucial to the accomplishment of these tasks (FitzGerald et al., 2016). As military leaders plan for future IT capabilities to foster technological innovation and efficiency within the DoN, they must address the fears that are often associated with OSS. A good starting point would be a review of the 2003 report conducted by The MITRE Corporation.

1. The MITRE Corporation's Study of OSS in the DoD

In 2003, the MITRE Corporation studied the role of Free OSS (FOSS) in the DoD. The MITRE study of government use and interest in OSS highlighted the importance of OSS applications in government software and provided an insight on how future government programs and applications may benefit from the open source community. According to the report, "the MITRE Corporation operates federally funded research and development centers (FFRDCs), which are unique organizations that assist the United States government with scientific research and analysis, development and acquisition, and systems engineering and integration" (MITRE Corp, 2003, p. 2).

MITRE determined that OSS plays a critical role in the DoD (MITRE Corp, 2003). MITRE reported that, "Over a two-week period, the survey identified a total of 115 OSS applications and 251 examples of their use" (MITRE Corp, 2003, p. 2). According to the MITRE report, "The word free in FOSS refers not to fiscal cost, but to the autonomy rights that FOSS grants its users. The phrase open source emphasizes the right of users to study, change, and improve the source code—that is, the detailed design—of FOSS applications" (MITRE Corp, 2003, p. 2). The autonomy rights granted to OSS users permit them to change code, rewrite it, distribute, and copy it as they see fit. These rights to control the software and its coding allows the DoD to integrate OSS in many programs and certain areas such as Infrastructure Support, Software Development, Security and Research as mentioned in the MITRE report (MITRE Corp, 2003). The four categories of infrastructure support, software development, security, and research will provide the "road map and support basis" to the scalable integration of OSS in the DoN (MITRE Corp, 2003, p. 17). The MITRE report suggests that banning its use in network applications may have negative consequences because it would limit the ability of the DoD to change infrastructure source code rapidly to adapt to changing cyberattacks. (MITRE Corp, 2003).

There are many costs associated with OSS that are not readily known. The MITRE report detailed the misconception that OSS is completely "free." The reality is, OSS software is not totally free; there are intangible costs (MITRE Corp, 2003). Although it may be cheaper to integrate in the beginning stages of implementation, the cost of using OSS could rise as use continues and maintenance is needed to maintain the software. Software sustainment costs will vary depending on the type of software, applications, and support that are required for functionality.

Since 2003, OSS applications in the DoD have increased in use and will only continue to increase as technology improves and develops. In fact, there is a direct correlation between the increase of OSS applications in the DoD and the reduction in long-term support cost (MITRE Corp, 2003). The MITRE Corporation noted that the OSS solutions that have large communities have a greater reduction in cost over long periods of time (MITRE Corp, 2003). In their research, the MITRE Corporation found that certain communities have many experts contributing to perfecting code, which leads to resolving problems faster than is typically done with the generalized support provided with proprietary software (MITRE Corp, 2003).

2. DoD Open Source Policies

A review of the history of DoD policy, statements, and memorandums regarding OSS presents a clear message. In an attempt to achieve faster, more secure, and cost-effective functioning software systems, OSS should be more routinely considered throughout various platforms within the DoD. In 2002, the DoN had established programs that contained OSS, however, it was not being integrated in many areas where OSS would be feasible (MITRE Corp, 2003). More guidance was required and the DoD Chief Information Officer (CIO) published a memo in 2003 that addressed and encouraged the use of OSS in the DoD. Still, OSS was not widely and positively perceived by many in the DoD. David Wennergren, author of the 2009 DoD memorandum titled "Clarifying guidance regarding OSS," noted that due to "misconceptions and misinterpretations of existing policies" the DoD CIO had to publish another memo in 2009 which superseded the 2003 memo (Wennergren, 2009, p. 2). The DoD CIO's 2009 memo stated that OSS can provide a military advantage and help the DoD "anticipate new threats and respond to continuously changing requirements," as well as help the DoD "update its software-based capabilities faster than ever" (Wennergren, 2009, p. 1).

After the initial publications were released, the DoD began to release more guidance on the need to enhance their IT capabilities. On 9 January 2010, Navy.mil released the article "Navy Stands Up Fleet Cyber Command, Reestablishes U.S. 10th Fleet," the article states, "the Fleet Cyber Command (FCC) and 10th Fleet were created as part of the CNO's vision to achieve the integration and innovation necessary for warfighting superiority across the full spectrum of military operations in the maritime, cyberspace, and information domains" (United States Navy Chief of Information, 2010.). More recently, during the March 2017 Executive Offsite event, the Commandant of the Marine Corps (CMC) directed that a Cyberspace Operations Occupational Field (OccFld) be created by the release of the February 2018 Authorized Strength Report (ASR). Marine Corps Administrative Message (MARADMIN) 164/18 stated that the "1700 OccFld will provide the Marine Corps with a deliberate, professionalized, and sustainable cyberspace workforce enabling the Marine Corps to conduct cyberspace operations, as directed by U.S. Cyberspace Command (USCYBERCOM)" (O'Donohue, 2018). These policies were published to foster an environment of IT growth, adaptability, and speed using OSS (Wennergren, 2009).

Years later, in 2016, the Office of Management and Budget (OMB) published a memorandum that highlighted current practices of creating but not sharing source code.

OMB concluded that the government was not being transparent and that challenges spawned from this practice "may result in duplicative acquisitions for substantially similar code and an inefficient use of taxpayer dollars" (Scott & Rung, 2016, p. 1). The OMB went on to encourage the use of OSS for benefits of cost saving to the tax payer, continual code improvements, better software peer review, increase knowledge sharing, and improved security testing (Scott & Rung, 2016).

The recent military memorandums and messages combined with current publications and policy set the foundation for OSS and have made the DoN environment ready for growth in the open source and IT realm. In addition to the policies, the DoD published the open source improvement cycle that outlines processes and procedures for service members and developers to work together with OSS. Figure 1 highlights the collaborative process of OSS by showing how the user can provide software improvements from the bottom level up to the developer (Department of Defense Chief Information Officer, n.d.).

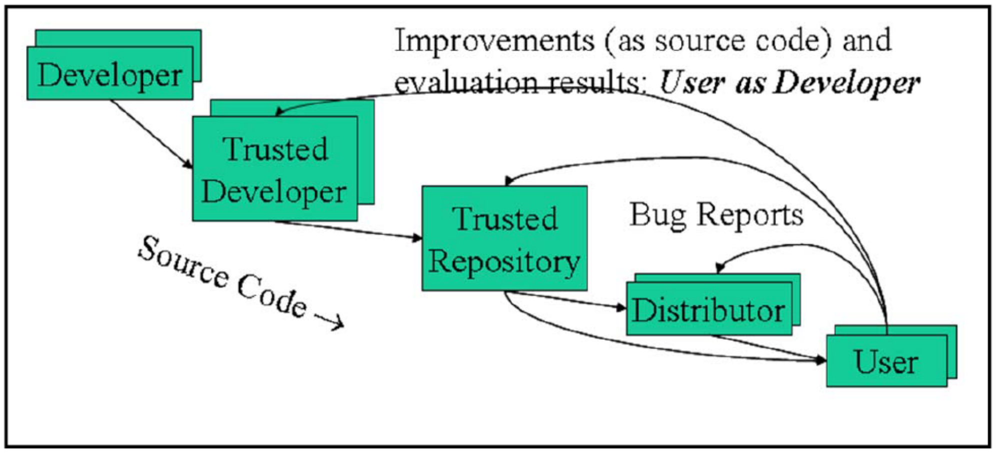


Figure 1. Open Source Improvement Cycle. Source: Department of Defense Chief Information Officer (n.d).

In today's war fighting environment, the DoN relies heavily on software systems to program and operate its many weapon systems, manpower accountability, and communication assets in order to gain a tactical advantage on the battlefield. Since the first major DoD OSS reports in 2003, the role of OSS has expanded to a variety of fields including database management, computing infrastructure, modeling and simulation, and many more (Koltun, 2011). In addition to these fields, OSS has already begun to be integrated throughout DoN as well as all DoD software systems. According to the DoD CIO website,

> Today, many DoD software systems are combined with “Linux distributions” which provide suites of such software as Red Hat Enterprise Linux, Fedora, Novell SuSE, Debian and Ubuntu. Other OSS implementations of Unix including Solaris, OpenBSD, NetBSD, and FreeBSD are also being successfully used in DoD systems. (Officer U.S. Department of Defense Chief Information Officer, 2018, p. 1)

B. OPEN SOURCE TRAITS

Currently, many applications in the DoN run on proprietary software. Using proprietary software has traditionally been standard operating procedure for the DoN and, as such, is the comfortable and known option for program managers and military leadership. This section will explore OSS through discussing and comparing OSS and proprietary software, highlighting classifications of OSS as COTS, and raising security concerns commonly found among OSS opponents. Most proprietary software applications consist of closed-source coding, which does not allow the user to freely manipulate the code to best serve the user as needed. In fact, many old or outdated versions of a DoN system, commonly referred to as “legacy systems,” and desktop applications are run on and are compatible with Microsoft/Windows operating systems (Shachtman, 2010). This might serve to be a problematic issue in terms of total functionality and interoperability between OSS and proprietary or closed-source applications. Establishing the use of OSS and ensuring all regulations and functionality will not be simple; however, the benefit of OSS has the potential to be enormous for DoN in terms of functionality, customizability, and control of IT. According to the Center for a New American Security, **“**The DoD should instead seize this opportunity to make greater use of open source methods and more fully embrace the use of OSS. In doing so, it will gain the common mode benefits of open source platforms and methods, as well as important advantages specific to the DoD's needs" (FitzGerald et al., 2016). OSS has been tried and the benefits are competitive with alternative software options. (Center for Strategic and International Studies, 2007).

1. OSS as COTS

As OSS continued to grow exponentially, not only in the civilian sector, but also within the DoD, purchasing and licensing agreements of OSS began to meet barriers in the DoD acquisition pipeline that made it difficult for DoD to continue its purchase and use of OSS. Initially, the confusion of processing OSS into the DoD was due to the definition interpretation. According to Carey, who wrote the DoN OSS Guidance,

> The misconception that OSS is neither a commercial off-the-shelf (COTS) nor government off-the-shelf (GOTS) solution has hindered the DoN's ability to leverage the benefits of OSS methodology. Because of this misconception, OSS has not received equal consideration during the software acquisition process. (Carey, 2007, p. 1)

In order to streamline the acquisition process and implementation of OSS throughout the DoD and DoN, Carey issued that all OSS will be treated as COTS products. Therefore, by establishing a baseline and streamlining the acquisition process, the DoN was able to purchase OSS properly through the acquisition pipeline in order to better support IT in the DoN.

In order to meet the DoD's goals of net-centricity and interoperability, the DoD established that OSS would be treated as a COTS product (Carey, 2007). As such, OSS would adhere to all federal regulations, policies, and guidelines that apply to COTS products (Carey, 2007). This decision provided a baseline for establishing OSS but new policy and licensing agreements were still needed to align OSS with federal regulations.

As the DoD set policies to ensure that OSS met standards according to military policy, the DoN CIO also expressed the need to be in full compliance with OSS license agreements. The importance of obtaining and following licensing agreements is imperative to the DoN user or programmer that is going to need to alter the source code of the software. Even if DoN programs and software users are not interested in the modifying the code—and instead use the applications at its most basic functioning—they are still required to adhere to the license stipulations. There are many different types of licenses but generally, OSS can be grouped into three categories: permissive, strongly protective, weakly protective. The details of these license categories are explained in Table 1.

Table 1. Open Source License Categories. Adapted from Department of Defense Chief Information Officer (n.d).

License Category	
Permissive	These licenses permit the software to become proprietary (i.e., not OSS). This includes the MIT license and the revised BSD license. The Apache 2.0 license is also a popular license in this category; note that the Apache 2.0 license is compatible with GPL version 3, but not with GPL version 2.
Strongly Protective (aka strong copyleft)	These licenses prevent the software from becoming proprietary, and instead enforce a "share and share alike" approach. In such licenses, if you give someone a binary of the program, you are obligated to give them the source code (perhaps upon request) under the same terms. This includes the most popular FLOSS license, the GNU General Public License (GPL). There are two versions of the GPL in common use today: the older version 2, and the newer version 3.
Weakly Protective (aka strong copyleft)	These licenses are a compromise between permissive and strongly protective licenses. These prevent the software component (often a software library) from becoming proprietary, yet permit it to be part of a larger proprietary program. The GNU Lesser General Public License (LGPL) is the most popular such license, and there are two versions in common use: the older version 2.1 and newer version 3. An alternative approach is to use the GPL plus a GPL linking exception term (such as the "Classpath exception").

All personnel writing code or involved in the security or acquisition process for any OSS related item must have basic knowledge of OSS licenses. This pertains to all of the uses the DoN has for open source, including but not limited to; code writing and development, to project collaboration and management, with private sector companies on different software projects. The main categories of DoD OSS users identified in the MITRE Corporation's report are described in Table 2.

Table 2. Categories of DoD OSS users. Adapted from MITRE Corp (2003).

Category	Description
Scripting and basic code development users	Large category. This includes language and scripting applications such as Perl, GCC, bash and JBoss to write simple scripts and code packages
Advanced code development users	Medium category. Included are cases where large complex library routines (e.g., scientific and parallel processing routine) need to be incorporated into new software
OSS sponsors	Smallest category. Consists of those that had explicitly decided to use an OSS model to promote non-DoD development work on their project

2. OSS Information Security

A common theme among opponents of OSS is security. There are those who argue that OSS is vulnerable simply because the code is exposed. Others are curious as to how to keep trade secrets and classified information secure when everyone can see the inner workings of a system. Thus, cyber security and software integration are major concerns when using OSS. If anyone can access and modify software and code, then how will the DoN safeguard against vulnerabilities. Does OSS even meet security requirements? How will the DoN achieve standard implementation and integration of programs or software if everyone is operating on their own version of the code? All of these legitimate questions must be considered and addressed when considering an open source option. The bottom line is that OSS is approved and being used in many government areas. Due to its already deeply imbedded roots in DoD software, information security within the DoD simply cannot function without OSS (MITRE Corp, 2003).

Along with coding efforts and software integration, OSS is ingrained in DoD security as well. OSS contributes to these efforts in two ways,

> First, it has produced infrastructure software such as OpenBSD with low rates of software failure combined with early and rapid closure of security holes, which makes such systems useful as the security linchpins in broader security strategies.
>
> Secondly, the OSS communities have had a long-term fascination with developing more and more sophisticated applications for identifying and analyzing security holes in networks and computers, resulting in OSS products such as SARA (Security Auditor's Research Assistant) and Snort (multi-platform, lightweight, rule-based tool for detecting hostile intrusions into a network) that are invaluable to in-depth analyses of security risks. (MITRE Corp, 2003, p. 20)

Although there is always concern for security, the previous examples display two of the many ways and different variations that OSS contributes to the security of our software programs. Security is a concern of every DoD official and is required to be considered at every aspect of any software integration project.

3. Understanding Linux

In order to further understand OSS and the many traits that are associated with it, one must be aware of perhaps the most prominent open source operating system: Linux. Linux is referenced in several of the examples and case studies that are discussed in this research; thus, a broad understanding of Linux will enhance the reader's understanding of the topics. The name Linux is often referenced when discussing OSS. Linux is a free open source operating system created by Linus Torvalds. This operating system, often referred to as the Linux Kernel, is one of the most widely used open source projects in the world (Garrison, 2010). Basically, the Linux kernel allows users a common method of controlling computer processes and hardware. Linux has revolutionized the computer industry and is now used by many well-known companies such as Amazon, Google, and Facebook (McMillian, R. 2012). Jeremy Allison, Google engineer and lead developer on the Samba project said,

> More than 8,000 developers have contributed to the Linux kernel in the past seven years, according to the Linux Foundation. And it has even

> become a standard operating system on custom-built consumer devices. You can find it on everything from inflight entertainment systems to streaming video players to Google's Android phones. It became the plumbing. (McMillian, 2012, p. 2)

The "plumbing" is now being used in various methods and across multiple platforms. Linux comprises the roots of the strong foundation of OSS. One reason Linux has been such a success is because of the cost, more specifically the cost to obtain a license. The GNU General Public License allows users to access software and ensures that they share their work. This affords a countless number of code developers to view, test, and ensure the functionality of published software. Like the notion of survival of the fittest, this process naturally highlights the software that works the best because it is that software that is being most used. This process is continuous and occurs all the while engineers and developers are simultaneously viewing, testing, and improving the functionality of this software.

C. OSS PLANNING CONSIDERATIONS FOR DOD

More recently, the DoD established a collaboration project with the civilian sector and many open source experts, the program was introduced as the Code.mil. program. According to the DoD's 2017 news release NR-077-17, "The Code.mil program is an open source initiative that allows software developers around the world to collaborate on unclassified code written by federal employees in support of DoD projects" (Department of Defense, 2017). DoD is currently working with GitHub, a development platform for hosting, building and reviewing code, to examine a collaboration effort between experienced software developers and government employees in the IT field (Department of Defense, 2017). The website code.mil helps users by directing them to a repository that houses source codes for a wide range of projects (Department of Defense, 2017). Users across the DoD can then review and make suggested changes, which streamlines the entire code writing process (Department of Defense, 2017).

Git is a program designed by Linus Torvalds, and is the heart of the GitHub website. Torvalds designed Git to facilitate the source code management of different developers creating, building, reviewing and changing software simultaneously with other developers. Warren reports “there are 85 million repositories hosted on GitHub, and 28 million developers contribute to them” (Warren, 2018, p. 1). According to the DoD, “The collaboration between the DoD and GitHub allows for Code.mil to create a network of peers between the federal government and the developer community to encourage participation, share knowledge, and make connections in support of DoD programs” (Department of Defense, 2017). The entire project allows for the design and building of critical software to meet the demands of today’s military force.

Overall, OSS has been an ambiguous concept to government planners. Although many OSS programs have been involved in government software for over 15 years, it has not been fully embraced due to several different occurring themes and concerns (Wennergren, 2009). Carey pointed out some misconception of the DoD guidance as being a potential barrier to adoption (Carey, 2007). In “OSS in Government: Challenges and Opportunities” the authors David A. Wheeler, Institute for Defense Analysis, and Tom Dunn, Georgia Tech Research Institute, provide an in-depth analysis that present key challenges and opportunities for a maximization of use for OSS within the Federal Government. By identifying these challenges, and with the use of interviews, Dunn and Wheeler were able to identify issues with the procurement, application, and embracement of OSS. They were also able to help provide recommendations to help resolve these issues. Through their research, proposed solutions, and recommendations Dunn and Wheeler have laid a foundation to assist the government in finding success in the use and application of OSS.

1. OSS Challenges and Opportunities Background

As with any new program involved with government integration, difficulties can come in many forms of obstacles. Dunn and Wheeler were able to identify many different issues as well as opportunities that OSS could present to government application. In their approach, they interviewed many different subject matter experts to help with their analysis and recommendations. According to the authors, “This document identifies key challenges and opportunities in the government application of OSS so that inappropriate roadblocks can be countered or mitigated. These challenges and opportunities were identified in interviews with experts, suppliers, and potential users, where users include both government contractors and government employees" (Dunn & Wheeler, 2013, p. 5). By going to the source experts, in not just OSS but in different areas of the entire chain management, Dunn and Wheeler provided details from every aspect in chain management that are affected by full integrations of OSS. In their research, Dunn and Wheeler "interviewed 31 people who were (1) OSS experts, (2) suppliers (especially non-government OSS suppliers), or (3) potential users (aka the demand side). The potential users included both government contractors and government employees (military and non-military, federal and non-federal)" (Dunn & Wheeler, 2013, p. 5). Table 3 displays the interviewees by category.

Table 3. OSS Challenges Interviewees. Adapted from Dunn and Wheeler (2017).

Category	Number
OSS Experts	7
OSS Suppliers	7
Contractors / Integrators	5
Government Employees	12
Total	31

Rather than conduct a quantitative survey, the interviewers took their time when letting the interviewees answer their questions and providing real feedback. The interview itself was guided so that the questions given about OSS were broad, leaving the interviewees to fill in the information needed without being guided to a certain endpoint.

During the course of the interviews conducted by Dunn and Wheeler, interviewees explained that OSS had already been in a considerable amount of use in government software systems. Other Interviewees felt the impact of cost was a driving factor for integration by explaining that OSS is much cheaper to operate than proprietary software, therefore as economic times cause for budget reductions OSS would be a better choice when cutting cost. According to the Office of Management and Budget (OMB), "Each year, the Federal Government spends more than $6 billion on software through more than 42,000 transactions, which results in a fragmented and inefficient marketplace" (Scott & Rung, 2016). Given the scale of that spending, it is understandable that the U.S., like other administrations around the world, is considering open-source software and open software standards as a way of saving money (Amirtha, 2016). As economic concerns of overspending continue to plague government information systems, open source presents a solution that is cheaper, more affordable, and equally capable software.

2. OSS Fear and Inertia

In government, some of the biggest factors when introducing something such as OSS are the idea of change and the risk involved with change. Change is something many in government feel impedes the way business is conducted. The motto "if it ain't broke, don't fix it" seems to impede the government's ability to provide better functioning software to its consumer. The ideology that the Dunn and Wheeler refer to as "risk aversion" is what keeps those in government feeling safe about developing their own code rather than using code written by someone else (Dunn & Wheeler, 2013).

In their research, Dunn and Wheeler assert, "OSS represents a change in the business model, and government entities are averse to risk. Any change is viewed as a risk" (Dunn & Wheeler, 2013, p. 7). Avoiding risk is what those in government feel they are doing by not embracing the many opportunities that OSS presents. Generally, many in the government fear that having openly available source code will make it easier to access government systems and "hack the code." Others are worried about compatibility, license issues, having someone to hold accountable in case of emergency, and product or software support (Serbu, 2018).

Another big concern about the use of OSS is the fear of malware being installed by another party when coding. While it is true that many individuals have access to the source code of any particular OSS, it is the same high visibility and code validation that allows for the detection of malware. The argument can be made that transparency is actually an attribute of OSS, unlike of unknown hidden code found in other types of software. In "Government policy toward OSS: An Overview," Dr. Robert Hahn discusses added advantages of OSS. According to Hahn,

> Open source can pay off on the demand side, too. Some users of software greatly value the option “to look under the hood” and to have the ability to make changes. Access to source code, for example, makes it possible for information technology professionals who maintain computer networks to tailor generic software to their specific needs and to debug software on the fly. (Hahn, 2009, p. 2)

The ability to view the source code can lead to the detection and over all protection of the software, subsequently more quality code can be produced for future software.

Not all OSS is safe or resistant to malware. So, organizations created measures that allow for the detection of such malware. According the article “OSS in Government, Challenges and Opportunities,” “There can be good and bad OSS, but there are metrics to figure out which is which. Another contractor said “[OSS] can assert higher quality in a more transparent way; they can show their source code is quite solid through the use of software quality assurance tools” (Dunn & Wheeler, 2013, p. 10). The metrics that are created for malware detection and quality assurance tools in OSS can be created in such a way to meet demands of government regulation and protection standards. Currently, there are many programs that allow for the capturing of metrics data such as Grafana, and other programs such as Prometheus that allow for the advanced system alerting and monitoring toolkit of OSS (Taylor, 2017). Figures 2 and 3 provide a visualization of monitoring dashboards for Prometheus and Grafana.

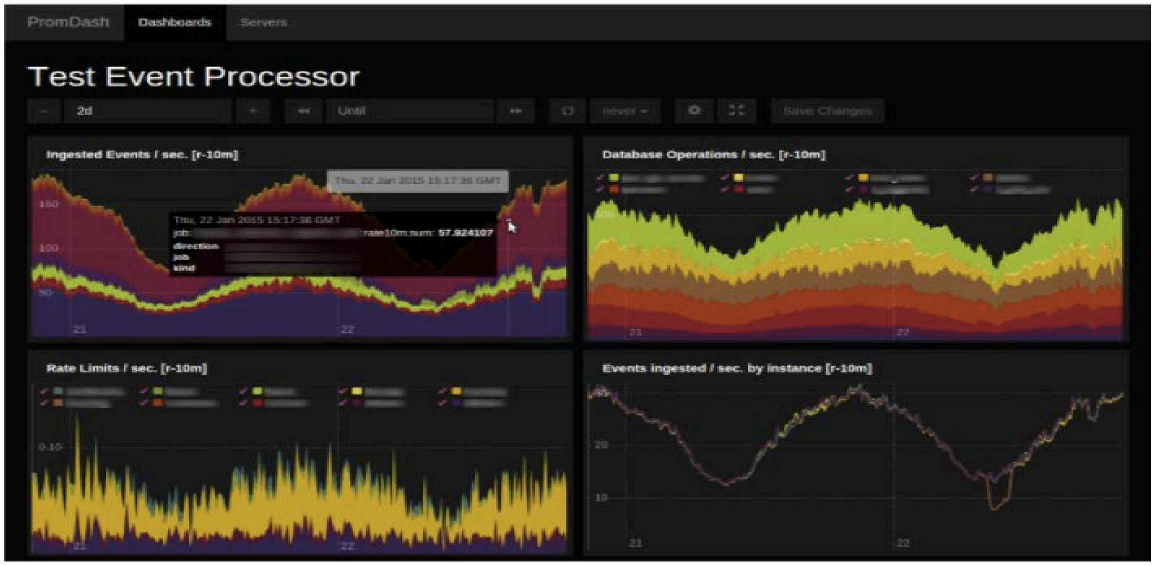

PromDash
Dashboards
Servers
Test Event Processor
Ingested Events / sec. [r-10m]
Database Operations / sec. [r-10m]
Rate Limits / sec. [r-10m]
Events ingested / sec. by instance [r-10m]
Save Changes

Figure 2. Prometheus Monitoring Dashboard. Source: Taylor (2017).

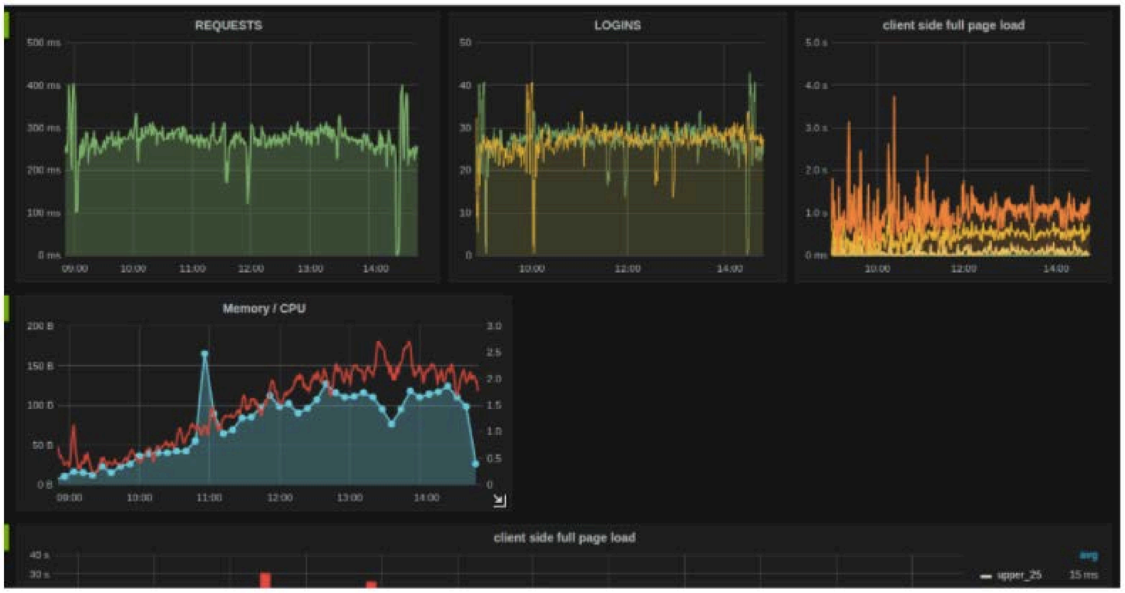


Figure 3. Grafana Monitoring Dashboard. Source: Taylor (2017).

Prometheus is a highly utilized open source-monitoring tool in the market today, and was initially created by Sound Cloud. Prometheus specializes in the monitoring of time-series data, alerts and auto-scale (Taylor, 2017). In Figure 2, the customized dashboard that is displayed for Prometheus shows the monitoring of multi-dimensional real-time data that Prometheus is able to provide. Although Prometheus is not a visual tool, like Grafana, the program is able to display the custom dashboard settings such as Ingested Events, Database Operations, Rate Limits, and Malware Alerting. Grafana, an open source visualization tool, is used by other applications (like Prometheus) to plot charts based on the time-series data as shown in Figure 3 (Taylor, 2017). Prometheus' main features are,

- A multi-dimensional data model.
- A flexible query language to leverage this dimensionality.
- No reliance on distributed storage; single server nodes are autonomous.
- Time series collection happens via a pull model over HTTP.
- Pushing time series is supported via an intermediary gateway.
- Targets are discovered via service discovery or static configuration.
- Multiple modes of graphing and dashboarding support. (Rabenstein, 2015, p. 2)

Software applications such as Prometheus that provide multi-dimensional data, scalable data collection, operational simplicity, and query language provide the government the ability to monitor OSS for malware.

3. Concerns with Warranties, Support, and Procurement

Reliability and customer support will always be concerns for organizations that are seeking new products or software. In government, the reliability of software support is one main reason why many government entities feel safe with proprietary software and hesitate when considering a switch to OSS. Therefore, when proprietary software is purchased through government contracts, the customer support is sold as part of the package. Consequently, if the customer or software support is unsatisfactory, there is limited ability to change customer support vendors; due to vendor lock-in. When it comes to service, OSS may present the better option because of the flexibility it gives the customer when choosing who provides support.

Another trait that OSS proponents tout over proprietary software is that it allows for the option to omit product warranty and customer support. With increasing budget restraints, some proprietary software sells a warranty along with its software. In some cases, the warranty may not be needed, but the warranty is part of the package therefore forcing government to pay for such warranties. According to Dunn and Wheeler, "Wasted money on commercial OSS support when it was available but known to be unnecessary. In one case, we were required to purchase a support contract for $30K [by management, even though there was no expectation that it was needed]" (Dunn & Wheeler, 2013, p. 12). The flexibility of using OSS, separate from vendor customer support, allows the government the ability to pick and choose which software capabilities can best support their needs. The DoD simply does not have to accept overpriced customer support that does not benefit their mission and slows operational tempo.

Incentivizing the acquisitions community to routinely consider OSS is a challenge. A government official observed, "It's an interesting problem—how do we change contracts to incentivize sharing? We need to change incentives to foster building a collaborative community and share code. I'm not sure we know how to do that. We could explicitly require past performance on how forthcoming they are on data rights and sharing code." (Dunn & Wheeler, 2013, p. 13). What the government official was referring to was the acquisitions process and the lack of monetary gain from sharing software vice contracting with one specific provider.

The acquisition process is set up to allow competition among companies for the government contracts. The issue many companies have with using code and software that can be "shared" is that there is very little incentive, or ability, to share secrets and possibly lose their bargaining advantage. By giving up its code to other OSS users, the incentive to produce software and become the main contractor for government may diminish. The monetary gains from outright owning a contract to now incentivized sharing can be significantly less for any contractor who is no longer the main provider for any particular contract. Therefore, how does the government create a system that allows for monetary gain through incentivized sharing? This could be accomplished by incentivizing contractors to share efforts in improving source code and creating better software for the DoD. Several recommendations were made by the authors on how to better the procurement process. They are as follows,

- Emphasize the value of accepting “80 percent solutions” and tailoring as necessary, instead of “100 percent solutions,” which have much larger costs and delays.
- Update processes to become faster and nimbler.
- Avoid imposing unnecessary paperwork burdens.
- Clearly state that source code must be shared within the government, as appropriate, if its development is government-funded.
- Consider switching to releasing unclassified software as OSS by default if its development is government-funded.
- Avoid presuming, when developing RFPs, that respondents will have a particular business model.
- Consider contributing Section 508 material (such as VPATs) for major OSS projects, to ensure that accessibility capabilities are documented.
- Include requirements in RFPs and contracts that the government must receive source code and unlimited rights if the development is government-funded, unless special waivers are granted.
- Consider switching to releasing unclassified software as OSS by default if its development is government-funded (Dunn & Wheeler, 2013, p. 17)

By considering the previous recommendations for improving the procurement process, government officials, specifically those in the acquisition community, may attain the ability to speed up the procurement process. This improvement to procurement will create a faster, more productive, and better incentivized process that leaves both customer and contractor satisfied (Dunn & Wheeler, 2013).

More recently, the government addressed the need of procuring custom developed software in the 2016 OMB Memorandum: “Federal Source Code Policy: Achieving Efficiency, Transparency, and Innovation through Reusable and OSS” (Scott & Rung, 2016). The initiative addressed the issue of existing software that cannot adequately satisfy the needs of the government. As a result, the memorandum encourages government agencies to consider acquiring OSS that will meet necessary government objectives (Scott & Rung, 2016). This particular memorandum helps to address procurement issues that were presented by Dunn and Wheeler. The memorandum also served as a beginning step to raise awareness in fostering more government agencies to consider OSS along with purchasing existing commercial software solutions to accomplish mission objectives.

4. OSS Education and Guidance

The lack of education will continue to be a barrier for open source until OSS training and education are mandated for specific decision makers in the DoD. This lack of education also leads to a continued reliance on the software producer to provide software support and guidance; allowing proprietary software producers to control software and support costs. With many OSS products already imbedded within many DoD software systems, the possibility of moving forward to a scalable open OSS architecture is not unfathomable. It is once again the knowledge gap that causes hesitation in government to fully embrace OSS. Therefore, education and guidance should be established in order to create a better understanding of how OSS works and the benefits it can provide.

In order to begin with the proper guidance and education, Dunn and Wheeler present several starting points. The starting points mentioned are as follows:

- **General OSS Education** - There is a need for general education about OSS, including the meaning of OSS and how it is developed, among both government employees and contractors.
- **Education on Intellectual rights** - There is a need for basic understanding of copyright laws and OSS. There is also a need for education and guidance on the implications of OSS licenses specifically.
- **Procurement Education** - There is a pressing need for education on OSS in procurement.
- **Need for guidance** - We recommend the government develop and release guidance on evaluating OSS (including the impact of OSS licenses such as the GPL), on contributing to OSS communities, and on releasing government-funded projects as OSS. (Dunn & Wheeler, 2013, p. 39)

As previously mentioned, there are several policies set forth by the DoD to help provide guidance and policy on OSS. By taking advantage of these starting points and further educating the government on use and implementation of OSS policy, standards can be set so that those individuals who are in the procurement process, education process, and leadership positions will possess the knowledge to increase the use of OSS within the government.

D. SUMMARY

The decision to fully embrace OSS into the Department of the Navy will not be easily made. Current regulations and policies have been established promoting OSS but still, obstacles persist. Leaders will have to these obstacles to create a future for OSS. Dunn and Wheeler conducted interviews and research that proposed several solutions to help create and guide the government in the use of OSS. According to Dunn and Wheeler,

> To maximally use its limited resources, the U.S. government must address these challenges and reduce the unnecessary barriers to the use and development of OSS. Many of these challenges can be addressed by promulgating education and guidance on OSS for different roles. The U.S. government should also transition to increased transparency and openness. (Dunn & Wheeler, 2013, p. 40)

These proposed solutions provide an excellent starting point for government and the ability to maximize the opportunities created from OSS. By adhering to the recommendations provided in the article "OSS in Government, Challenges and Opportunities," the government may find success in OSS, which may lead to a more efficient, cost effective, and better operating software needed for mission success in the future.

III. OSS BENEFITS AND VULNERABILITIES

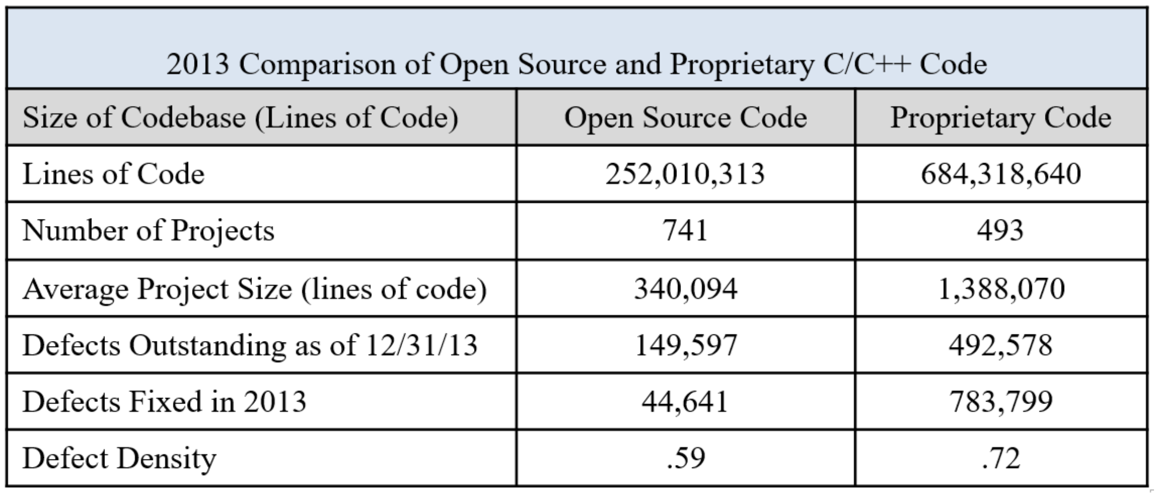
Organizations routinely seek ways to obtain and maintain efficiency in all aspects of conducting business. This can be accomplished in many different ways; however, one major consideration in any business strategy is cost. Unfortunately, the true "cost" of IT, specifically OSS, can be difficult to determine. One may favor OSS because of the cost savings of owning the code, with the ability to manipulate the code, versus purchasing outside software for an organization's software capabilities. OSS can also be measured in terms of time needed to acquire and implement an open source project or the personnel needed to develop and support software using open source code. Furthermore, OSS cost can be viewed from the perspective of code stability and interoperability (Active State, 2016). A deeper dive into the true cost of OSS appeals to business strategist because it allows an organization to use software with free software licenses, own the code, reduce the initial cost of project and product development, and also shortens the code writing or procurement process (Active State, 2016). This is accomplished by a few methods that are unique to OSS.

First, OSS uses a collective community to produce, screen and verify the software. Software produced in this manner is tested on multiple platforms, by multiple users. As a community of external users write, develop, and share the software, a company is essentially receiving free labor, production, troubleshooting, and testing of code. All of this will be accomplished with a large community with different experiences and diverse ideas (Bromhead, 2017).

This large talent pool, all working on open source products, will yield more reliable and secure software. Furthermore, this process allows for the "cream to rise to the top." Meaning that the products that actually work best, vice only in theory, will be used more, more readily available, known, more heavily vetted, and more trustworthy. The DoN can leverage this theory by attaining vetted OSS and applying it in useful areas. Forbes conducted "The Future of Open Source" surveys from 2011- 2013 and the top reasons why organizations sought to use OSS were "better quality service" and "freedom from vendor lock-in" (Sabhlok, 2013, p.1). Open source is considered to have better quality service because of the technical expertise and high efficiency levels that come from the large pool of contributors (MITRE Corp, 2001).

Despite this claim, vulnerabilities and defects persist in OSS. A company called Coverity that specializes in examining software for quality and security conducted a study in 2013 and determined that, "OSS had fewer defects in the code than proprietary software," as shown in Table 4 (Vaughan-Nichols, 2014c, p. 1).

Table 4. Comparison of Open Source and Proprietary Code.
Source: Coverity (2014).



2013 Comparison of Open Source and Proprietary C/C++ Code

Size of Codebase (Lines of Code)	Open Source Code	Proprietary Code
Lines of Code	252,010,313	684,318,640
Number of Projects	741	493
Average Project Size (lines of code)	340,094	1,388,070
Defects Outstanding as of 12/31/13	149,597	492,578
Defects Fixed in 2013	44,641	783,799
Defect Density	.59	.72

In addition to the fewer recorded defects, Coverity also found that defects found in OSS were being fixed faster than defects found in proprietary code (Coverity, 2014). Programmers using Linux were able to reduce the average defect fix time from 122 days to six days (Vaughan-Nichols, 2014c). Organizations deciding whether to implement OSS must consider the vulnerabilities that come with this particular type of software and how those vulnerabilities are addressed.

Vendor lock-in for software is another concern that OSS allows an organization to avoid. The lock-in comes with need for vendors to keep the details of their proprietary software a secret from their competitors which creates barriers to interoperability and flexibility. Vendors then seek to establish long-term contracts that not only consist of the software, but also training and support. The inability to see or manipulate any software used by an organization will limit internal troubleshooting capabilities and the ability to see the extras or vulnerabilities that may come with proprietary software (Bromhead, 2017). Some organizations will not be concerned with avoiding vendor lock-in because they are either incapable of customizing their software or their strategy does not gain from the ability to customize their IT experience (O'Connor, Ong, Sander, & Ferlo, n.d.). In some cases, OSS can lead to vendor lock-in by securing external contractors to support or "fill the gaps" of internal DoD capabilities.

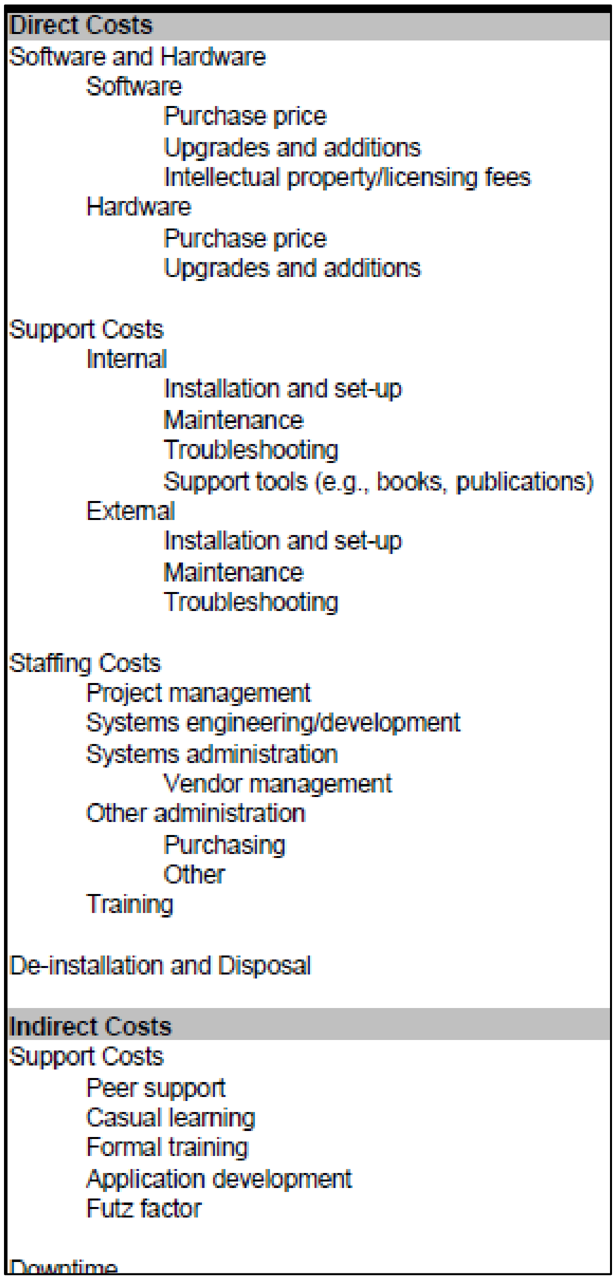
The DoN will have to examine each area of potential OSS implementation and determine if vendor lock-in would be a benefit or detriment, and how much risk they would be willing to accept for each program. In certain cases, avoiding vendor lock-in is financially beneficial. The Electronic Government Initiative in San Paulo, Brazil organized 72 telecenters that provided free Internet and computer use (O'Connor et al., n.d.). These centers were established using OSS and cost approximately $10,000 per telecenter. In this situation the initiative was able to establish the telecenters cheaper by using OSS than by contracting with a vendor (O'Connor et al., n.d.).

A. MEASURING THE COSTS OF OSS

Ultimately, organizations such as the DoN need to consider two types of costs: direct and indirect costs. Both of these types of costs have varying benefits and have different advantages and disadvantages in terms of performance. In addition to these overt measurable variables, there are also intangible variables such as peer support, usability, practicality, and scalability that must be considered (MITRE Corp, 2001). Direct costs are easily explained and relatively transparent when calculating and carefully considering everything that has an allocation of funds. Indirect costs are more complicated to calculate and will require heavy scrutiny when deciding if using OSS is more cost effective than using a COTS system. Program managers must consider the costs of reliability, functionality, and scalability and what these indirect costs mean to their programs (MITRE Corp, 2001). In order to help quantify both direct and indirect cost the MITRE report completed in 2001, developed a cost element taxonomy.

The taxonomy found in Figure 4 was designed to assist program managers make decisions on purchasing software while having a visual representation of the business and economic implications of their decisions. Included in this taxonomy is the “Futz Factor.” The Futz Factor is a term coined for time that is lost due to inefficiencies. An example of this is seen in the extensive amounts of time spend on polishing appearance rather than perfecting optimal performance (Moore & Stanton, 2018).

The report also provided a taxonomy of risks and benefits with an example rating system to address some of the intangible costs that come along with software. Every business decision will have business effects and additional business consequences. Given the knowledge of these effects, it would be wise to attempt to identify and, if possible, quantify the risks and benefits of a decision such as to allow scalable integration of OSS.



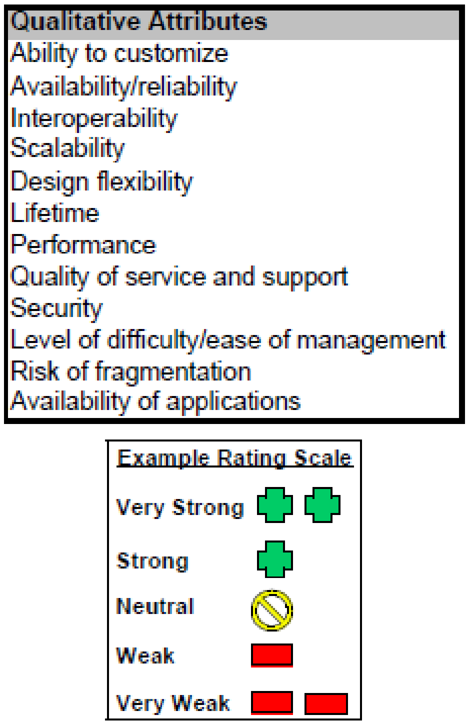
Direct Costs

- Software and Hardware
  - Software
    - Purchase price
    - Upgrades and additions
    - Intellectual property/licensing fees
  - Hardware
    - Purchase price
    - Upgrades and additions
- Support Costs
  - Internal
    - Installation and set-up
    - Maintenance
    - Troubleshooting
    - Support tools (e.g., books, publications)
  - External
    - Installation and set-up
    - Maintenance
    - Troubleshooting
- Staffing Costs
  - Project management
  - Systems engineering/development
  - Systems administration
    - Vendor management
  - Other administration
    - Purchasing
    - Other
  - Training
- De-installation and Disposal

Indirect Costs

- Support Costs
  - Peer support
  - Casual learning
  - Formal training
  - Application development
  - Futz factor
- Downtime

Figure 4. OSS Cost Element Taxonomy. Source MITRE Corp (2001).



Qualitative Attributes
Ability to customize
Availability/reliability
Interoperability
Scalability
Design flexibility
Lifetime
Performance
Quality of service and support
Security
Level of difficulty/ease of management
Risk of fragmentation
Availability of applications

Example Rating Scale	
Very Strong	✚ ✚
Strong	✚
Neutral	⊘
Weak	🟥
Very Weak	🟥 🟥

Figure 5. OSS Taxonomy of Risks and Benefits.
Source: MITRE Corp, (2001).

The idea behind Figure 5 is that it lists qualitative attributes that should be assessed when planning a software strategy. The DoN should consider and compare the strengths and weaknesses, from a qualitative perspective, for both OSS and other COTS options (MITRE Corp, 2001).

1. Cost Case Studies

The DoD spends billions of dollars on IT software and programs. These programs often come with complex contracts, which include both money and time commitments. In a 2016 memo, the Office of Management and Budget (OMB) highlighted the importance of saving taxpayer dollars and exploring OSS as a viable IT option (Scott & Rung, 2016). The DoN is not immune to high spending for IT systems and although OSS may not be the final solution, it may be able to help reduce costs. At the very least, it is important to look at the DoN's current programs and determine if OSS can be implemented to help reduce cost; in accordance with the 2016 OMB memorandum.

All of these costs can equate up to billions of dollars over the life cycle of a system or contract. In 2010, for instance, the Navy sent $3.3 billion to Hewlett-Packard (HP) for services, licenses, hardware, and software just to start weaning itself from HP services (Shachtman, 2010). The idea behind the Navy Marine Corps Intranet (NMCI) was to combine all of the various DoN systems into a single network and for all intents and purposes, NMCI was able to achieve that goal. Systems and networks were consolidated and functional, although the extensive policies and centrally managed network had a plethora of issues. This contract served as a vendor lock-in example, where the DoN paid nearly $10 billion over 10 years for a system that was riddled with bugs and lengthy downtimes (Shachtman 2010). Another example of high spending and the inability to make system adjustments to provide efficiency is the Defense Travel System (United States Digital Service, 2016).

The Defense Travel System (DTS) is the software that the DoD uses to provide travel arrangements. The software was implemented in 1998 with program development cost of $263.7 million (Perera, 2007). This system is linked with employees' government travel card (GTC) and can perform many functions to facilitate government employee business travel; including the ability to books flights, rental cars, hotel reservations, etc. DTS system has been in operation for more than a decade and is still encountering problems in the use, employment, and cost of operation. Initially, the Government Accountability Office (GAO) published a report that estimated approximately $56 million in savings, as seen in Table 5.

Table 5. Summary of DTS Estimated Annual Net Savings Reported in the September 2003 Economic Analysis. Adapted from Williams and Rhodes (2006).

Constant fiscal year 2003 dollars in millions	
Cost components	**Estimated annual net savings**
Records management	$19.8
Centrally billed accounts	1.7
CTO acquisition and administration	2.4
CTO services	31.0
Voucher process and compute	54.1
Voucher pay	0
Legacy systems	14.5
PMO	(8.8)
Help desk/DTA	(36.8)
System operations	(21.5)
Total net savings	**$56.4**

This report considered the DTS estimated costs of $2.1 billion from fiscal years 2003–2016 (Williams & Rhodes, 2006). Later, in a 2016 report to congress the Government Accountability Office (GAO) determined, “that DoD had overestimated savings for DTS and failed to fix implementation problems with the system nearly a decade ago, DTS added fees for the user and prevented travelers from quickly making changes to their reservations” (United States Digital Service, 2016). Overall, the DTS handles about $3.5 billion annually for travel accommodations with a per transaction cost of approximately $10 (United States Digital Service, 2016). Each transaction for this GAO report refers to the exchange of money from arrangements made through DTS.

Ultimately, DTS has been a largely successful project in terms of arranging travel. Government employees routinely use the service to conduct official travel and generally accomplish their desired travel requirements. However, when accounting for the actual cost of operating and then add the qualitative costs, risks, and benefits from Figure 5, the program is still lacking. After pouring billions of dollars into the program, the system still has a variety of unresolved issues that seem to have eluded the government since 1998.

2. Forge.mil

Given the high spending cases of NMCI and DTS, one must consider the questions, “Could a cheaper OSS option be implemented that meets all organizational needs, user needs, and meets security requirements, or how can OSS components help to lower TCO?” The website forge.mil was established to explore questions like this. Forge.mil is a site that was created to prevent project teams from “re-inventing the wheel,” as discussed by the 2016 OMB report. The DoD has multiple layers of software development that is often times inconsistent and incompatible with other related software (Nimmer, 2015). Programs and software are routinely developed in an ad-hoc manner or in silos, which cause problems when interacting with other systems or performing mission related interoperable functions (Nimmer, 2015). The goals of forge.mil were,

> to create a more open and transparent development process that could remove barriers to reuse, encourage collaboration, and discourage proprietary or closed systems. Build such an extensive, collaborative community required a powerful and adaptable Application Life Cycle Management (ALM) platform to enable code reuse and quality improvements, as well as improve of time to market for new applications. (Nimmer, 2015, p. 1)

The use of forge.mil resulted in both tangible and intangible benefits. The Defense Information Systems Agency (DISA) recorded savings of approximately $18,000 for small projects $1.2 million per project for large teams. Furthermore, the establishment of forge.mil created a large open source community of developers and projects. In 2015, forge.mil, recorded “24,000 registered users, 900 projects, 200 active groups, 2,900+ applications and 150,000+ downloads.” (Nimmer, 2015).

Forge.mil is composed of two different sections. The first is SoftwareForge.mil, which hosts for internal projects only and usually considered free software (Martin & Lippold, 2013). The second section of forge.mil is ProjectForge.mil, ProjectForge.mil was created for users who required their own private space due to limited access to property rights on other software; it also requires a fee from its users for the services provided (Martin & Lippold, 2013).

Since its inception the use of forge.mil has grown significantly in the DoD. Forge.mil has been implemented in many DoD programs, the case study of the Communications Electronics Command (CECOM) provides an example of forge.mil implementation within DoD systems. CECOM manages different aspects of software programs that are rooted in logistics, artillery, and other various military functions. Initially, CECOM project teams were siloed with limited ability to leverage the work of its adjacent teams. With project teams siloed they were unable to access common code. Therefore, each time a new project emerged, new teams would then have to reinvent their processes and procedures. CECEOM was plague by other issues such as, the lack of visibility on code and lacked the ability to track code according to its software life cycle. Forge.mil was chosen by CECOM mainly because it was a DoD implemented system. Once implemented, many of CECOMs software issues were resolved, the code was now tracked and visible to software engineers within the project teams. This allowed for faster start up times and improved code (International Data Corporation, 2013).

The case study of CECOM is one example of the forge.mil implementing OSS to achieve mission readiness, cut cost, and improve software life cycles for DoD software systems. Forge.mil host other projects as well and continues to gain popularity within the DoD. As software systems continue to need upgrades from legacy systems, platforms such as Forge.mil will no doubt be applicable to benefit certain projects in the DoD. With Forge.mil currently operating and recording cost savings on the government’s Non-classified Internet Protocol Router (NIPR) and Secure Internet Protocol Router (SIPR) networks, why does the DoN not more routinely seek to use OSS as a primary option for software and code development? Perhaps before that decision is made, the DoN should consider the total cost of ownership for OSS.

B. TOTAL COST OF OWNERSHIP

The Total Cost of Ownership (TCO) of using OSS must be considered when deciding between writing and developing code, choosing a vendor, or starting the acquisitions process for a project. This cost covers all of the additional costs that will be included when purchasing or developing software. Costs for using software include the software, training, support and legal licensing, maintenance, and development (ActiveState, 2016). Other considerations such as long-term support, maintaining the software or system over time, customizing, or enhancing software. Organizations or project managers will have to consider all of these variables when seeking to use OSS. Considerations such as these are illustrated in Table 6.

Table 6. Total Cost of Ownership Table. Adapted from ActiveState (2016).

Costs	OSS	Proprietary
Acquisition Cost (software licenses)	None	Potentially millions of dollars
Training	Developer salary * days training	Developer salary * days training + in-house time and employee salary * days training
Development	Developer salary * development months + fixed cost of in-house open source expert	Developer salary * development months + variable cost of external expert
Maintenance and Support	Full-time salary + fixed cost of in-house open source expert of consultant fee	Full-time salary + variable cost of external expert and recurring consultant fee
Legal (Distribution Rights and Indemnification)	Time for license audit/building governance process + potential license infringement risk costs	Time for license audit/building governance process + potential license infringement risk costs

As seen in Table 6, using OSS can save money, potentially millions of dollars in the upfront licensing and software acquisition costs. In the other cost categories, the proprietary option will have additional expenses that will only increase the total cost of proprietary software over OSS. As one examines Table 6, OSS looks like a better option than a closed-source or proprietary system, but different organizations have experienced varying levels of success after implementation. The California Department of Transportation spent approximately $220,000 on an open source identification and password security solution (O’Connor, Ong, Sander, & Ferlo, n.d.). Their Linux-based system worked as planned, reduced the total cost of ownership, and was approximately $280,000 cheaper than the proprietary alternative (O’Connor et al., n.d.). One the other hand, Mexico decided to implement an OSS option for 120,000 of their schools. The project leaders for the Red Escolar Libre project choose the OSS because they wanted to avoid paying for software licenses. Ultimately, the cost of support for their OSS began to skyrocket. Project leaders were forced to rethink their decision to use OSS and began the search to complete the project using a proprietary option (O'Connor et al., n.d.).

The final case study in this section occurred in Munich, Germany. This is an example that shows there is much more to OSS than monetary costs. Factors such as compatibility and interoperability can plague a system that does not have a flexible design. In 2001, the city of Munich Germany launched an initiative to move to an open source platform LIMUX (LINUX in Munich). The city conducted extensive research and determined that the city would benefit from OSS rather than proprietary software. The proprietary software was Microsoft. Until that point, the entire city operated under Windows platform (Windows NT) (Heath 2017). As the end of the contract drew close, Microsoft presented the upgrade to Windows XP. Instead, the city council sought to look at OSS for a possible alternative to Microsoft product. Many factors played a role in the decision from cost, to the local economy, to security concerns (Heath, 2017).

Upon their final analysis, the city council figured that their move would be strategic vice monetary (Saunders, 2014). Although numbers and calculations pointed to initial higher cost of transition due to new software, infrastructure, and hardware, the city still deemed OSS a better fit. They planned to design the code, using their own particular skills, in order to fit their own security concerns and change software as they saw fit. As time went on, the entire city was able to change their infrastructure to meet their needs for OSS. Munich became a beacon of hope and an example of perfectly executed OSS implementation for OSS advocates everywhere (Heath, 2017).

Later in 2017, it was determined that Munich would back track their OSS initiative and reequip their entire infrastructure to work, once again, with Microsoft (Heath, 2017). This move was a shock to the OSS community. From an outside perspective, it was hard to understand why Munich would make such an arduous move; especially after all of the work went into redesigning their entire IT infrastructure. The issue that still plagued Munich was that 4,163 Windows based computer still needed to operate on Microsoft Windows (Heath, 2017).

The downfall of Munich was simply the inoperability of the OSS LIMUX to work on Microsoft Office and vice versa. The city council in Munich also believed that in the long term, LIMUX would prove to be unstable and cost would increase dramatically. However, it was the rumblings across the business sectors in Munich who were complaining that businesses needed the Windows platform in order to not only operate with local Munich businesses, but global windows-based businesses as well (Heath, 2017). Too much avail, the city council decided that the idea of running two different types of systems independently would end up being too costly for local businesses and the taxpayer. Therefore, the entire city made the switch back to Microsoft platforms.

The Munich case likely failed due to interoperability issues with the Windows Platform. One option the city council could have looked at implementing, was ensuring that open standards was an option, which could have possibly helped ease their transition to operating with Microsoft products. Opensource.com defines open standards in the following manner, "Open standards act as a guideline to keep technologies open especially for open source developers" (Endsley, 2018, p. 1). Open standards allow the interoperability and allow applications to talk to each other (Heintzman, 2003). Recently, Gregg Brown, the Senior Director for the Interoperability Group at Microsoft, said, "Today, developers frequently decide to implement widely used open standards rather than design new protocols from scratch. That's certainly the trend at Microsoft" (Brown, 2011, p. 1). With Microsoft pursuing the use of OSS and open standards a possible solution could have been implemented for Munich. An open standards interface would have allowed Microsoft and LIMUX to exchange information and communicate with each other. This would have been a critical element to ensure that Munich could have continued on their path to using OSS rather than allowing interoperability issues to plague and discontinue their use of OSS. The Munich case serves as an example of why the DoD should consider a scaled approach to OSS. As discussed earlier in this research, not every type of software platform is going to be applicable to OSS.

These three examples show that the success of OSS, and whether or not OSS will decrease the TCO, depends on the organization. Organizations have different anatomy and various capabilities that should be weighed when dealing with OSS. The Mexico case is a good example of how OSS was not the best option in terms of cost. Research conducted by the Gartner Group showed that companies often spend only about 8% of their IT budget on software and 92% of the budget on other costs such as installation, support, subscription fees, training, and downtime (O'Connor et al., n.d.). These studies suggest that organizations should consider all aspects of cost, not only the licensing fees that can be saved by using OSS (O'Connor et al., n.d.).

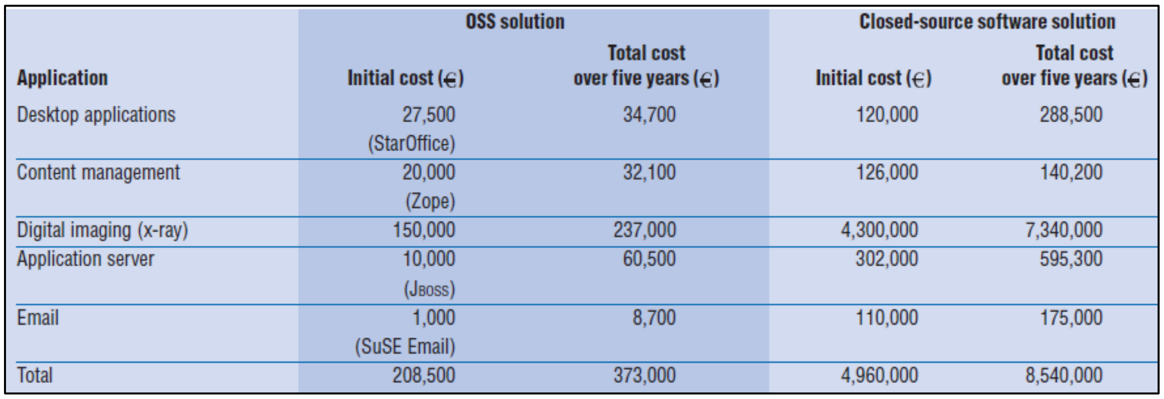
C. BEAUMONT HOSPITAL OPEN SOURCE CASE

Fitzgerald and Kenny conducted a study of a hospital in Ireland named Beaumont Hospital. The hospital employed over 3,000 employees and had a significant need for an upgrade in its IT systems. This upgrade would not only include back-office servers but front-office applications as well (Fitzgerald & Kenny, 2004). Once complete, Beaumont Hospital would undergo a complete IT upgrade using primarily OSS. The hospital decided to conduct this task in two phases. The first phase would review and select certain products for upgrade, which were mainly email and generic desktop applications. The second phase addressed operational systems that helped the organization conduct business on a daily basis (Fitzgerald & Kenny, 2004).

Beaumont upgraded its system in several areas. One area was the content management system, called Zope. Beaumont downloaded the product for free but spent roughly €20,000 in software support from a small support company. This service functioned as desired, allowing the hospital to plan meetings and events, tag people with alerts, procedures, and events, and linked other hospital systems and servers for a more interoperable, inter-domain, and customizable program (Fitzgerald & Kenny, 2004). Systems were "talking" to each other, which increased efficiency in the hospital. Additionally, Beaumont was trying to implement an open source system called Veteran's Health Information Systems and Technology Architecture (VistA). VistA is a proven open source system that has been in operation by the U. S. Department of Veterans Affairs and the U.S. DoD for more than 20 years (Fitzgerald & Kenny, 2004). The federal government has successfully thrived on this open source system, which is supported by an extensive network of programmers. In addition to the U.S. DoD, VistA is also used by countries like Finland, Germany, and Nigeria (Fitzgerald & Kenny, 2004). This is an example of OSS being used worldwide while handling secure medical information.

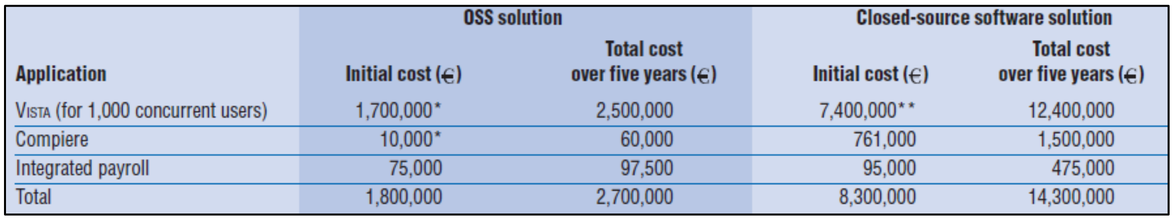
In the end, at Beaumont Hospital, the two-phase upgrades revealed major savings during the implementation of the new open source IT solution. Tables 7 and 8 show the results of phase one and two respectively.

Table 7. Beaumont Hospital Phase 1 Solutions.
Source: Fitzgerald and Kenny (2004).



	OSS solution		Closed-source software solution	
Application	Initial cost (€)	Total cost over five years (€)	Initial cost (€)	Total cost over five years (€)
Desktop applications	27,500 (StarOffice)	34,700	120,000	288,500
Content management	20,000 (Zope)	32,100	126,000	140,200
Digital imaging (x-ray)	150,000	237,000	4,300,000	7,340,000
Application server	10,000 (JBoss)	60,500	302,000	595,300
Email	1,000 (SuSE Email)	8,700	110,000	175,000
Total	208,500	373,000	4,960,000	8,540,000

Table 8. Beaumont Hospital Phase 2 Solutions.
Source: Fitzgerald and Kenny (2004).



	OSS solution		Closed-source software solution	
Application	Initial cost (€)	Total cost over five years (€)	Initial cost (€)	Total cost over five years (€)
VISTA (for 1,000 concurrent users)	1,700,000*	2,500,000	7,400,000**	12,400,000
Compiere	10,000*	60,000	761,000	1,500,000
Integrated payroll	75,000	97,500	95,000	475,000
Total	1,800,000	2,700,000	8,300,000	14,300,000

The savings in phase 1 resulted in favor of the OSS solution. Savings were approximately €4.7 million and €8.2 million, when applied over a period of five years (Fitzgerald & Kenny, 2004). In phase 2, the initial cost savings were approximately €6.5 million, and the total five year cost savings were €12 million (Fitzgerald & Kenny, 2004).

The case study of Beaumont Hospital saw significant savings by using OSS solutions. Beaumont choose open source options for the majority of their programs but still had some proprietary software and systems in use. They found that OSS allowed the hospital to continue performing all functions while bringing down their total cost of ownership (Fitzgerald & Kenny, 2004). This is an example of OSS still being beneficial and operational even though all systems were not using OSS components. This case supports the idea that OSS can be used for only certain functions and in specific areas of an organization. Beaumont did not force an OSS implementation on all of their programs and systems because there were specific areas in which a non-open source option would have been more efficient, reliable, and resilient. The Beaumont case may serve to be a good example of how the DoN could consider using OSS. Leaders can use OSS components in various aspects of the IT strategy, still saving on cost in certain areas, while understanding that there will be areas and times when a proprietary option is better to meet mission needs and requirements.

D. OSS VULNERABILITIES

OSS has many benefits and proponents often tout the community of contributors as the ultimate example of checks and balances, however there are examples of bugs and vulnerabilities. In 2014, an OSS zero-day vulnerability called "Heartbleed" that was discovered (Vaughan-Nichols, 2014a). This was a major vulnerability that would allow hackers undetected access to data and secure servers. The 2017 Coverity Scan Report describes Heartbleed as follows, "The Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable version of the OpenSSL software" (Llaguno, 2017, p. 4). Heartbleed was found to be a programming problem and was patched less than 24 hours of public discovery, yet it still affected hundreds of millions of websites (Vaughan-Nichols, 2014b). The 2017 Coverity report indicated that a positive outcome of Heartbleed was the maturation of the community (Llaguno, 2017). By maturing, Coverity means that software developers are now conducting more frequent analysis of their products with scanning software (Llaguno, 2017). In 2016, Coverity recorded 4,117 active open source projects were submitted to be scanned for vulnerabilities, 50% using Travis CI (Llaguno, 2017). Wikipedia describes Travis CI as "a hosted, distributed, continuous integration service used to build and test software projects hosted at GitHub. Open source projects may be tested at no charge via travis-ci.org" (Llaguno, 2017).

Perhaps a more well-known intrusion was the 2018 Equifax breach. Equifax has components of OSS and, despite the security advantages that OSS offers, the company still lost positive control of its data (Korolov, 2018). These examples prove OSS is not impenetrable; however, enterprises are still turning to OSS that offers them more tools and agile methodologies to accomplish organizational goals (Korolov, 2018). Maria Korolov, a 20-year emerging technology writer and current contributor for CSOonline, reported that 96% of commercial applications have open source components in them. Korolov continues to state, "The average application had 147 different open source components—and 67 percent of the applications used components with known vulnerabilities" (Korolov, 2018). A 2017 report by an organization called Snyk outlined the increasing vulnerabilities generally found in OSS, as seen in Figure 6.

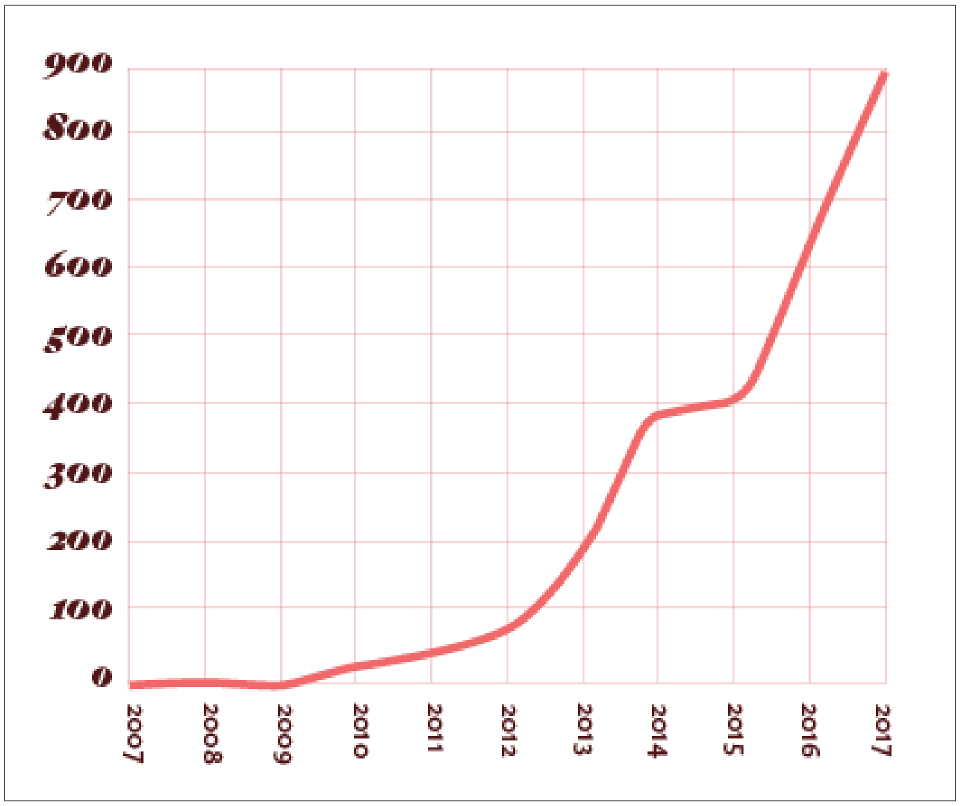


Figure 6. Published Open Source Vulnerabilities by Year.
Source: Snyk (2017).

There could be a myriad of reasons for the reported increase in vulnerabilities over time, but the point is that OSS vulnerabilities exist. Snyk monitored more than 430,000 sites with open source components and continues to see an increase in open source vulnerabilities (Snyk, 2017). In 2016 alone, there was a 53% increase in the number of published vulnerabilities found the OSS surveyed sites (Snyk, 2017). After its general report, Synk choose to look at a specific organization and found different results. Snyk isolated the results for Red Hat Linux and found a decrease in vulnerabilities over time.

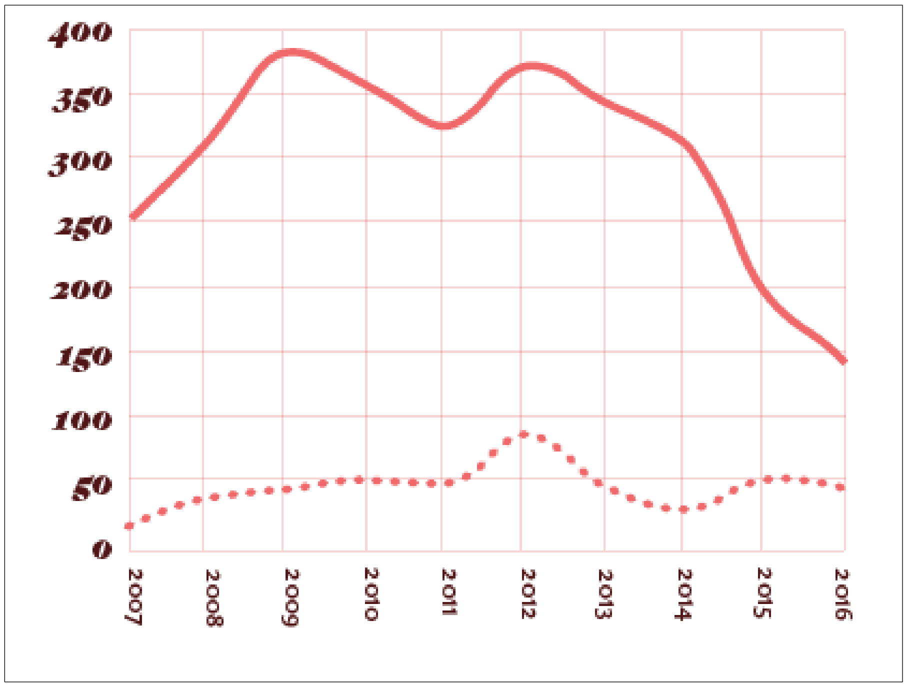


Figure 7. Red Hat Linux Vulnerabilities by Year.
Source: Snyk (2017).

Figure 7 shows a downward trend in vulnerabilities, with the solid line representing regular vulnerabilities and the dotted-line representing critical vulnerabilities. Overall, Red Hat has seen a 62% decrease in vulnerabilities since 2012 (Snyk, 2017). The takeaway is that not all OSS is equal in terms of vulnerabilities, security, and support. If organizations choose to use OSS, they must conduct thorough research and find a reliable option.

The Snyk report continued its research into OSS by looking at vulnerabilities and how long it takes to fix those vulnerabilities. They found that, as mentioned earlier, the OSS community has a large number of contributors who are willing and eager to fix vulnerabilities (Snyk, 2017). In Red Hat Linux, 69% of vulnerabilities were fixed within a day and 90% of vulnerabilities were fixed within two weeks of public disclosure (Snyk, 2017). The general report, as seen in Figure 8, on OSS was similar to Red Hat, showing maintainers could respond to 94% of OSS vulnerabilities within a week of public disclosure (Snyk, 2017).

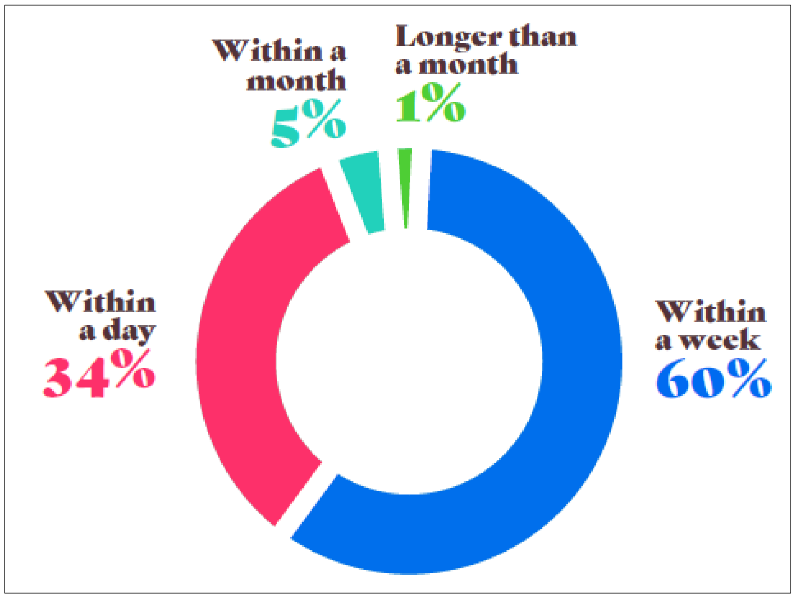


Figure 8. How Quickly Maintainers could Respond to a Vulnerability. Source: Snyk (2017).

The bottom-line is that OSS, like all software, has vulnerabilities. Any organization that chooses to use OSS in any capacity needs to understand that vulnerabilities exist and that maintenance will be required on all types software and systems. Organizations have learned that system upkeep and monitoring are still important; even with OSS. Understanding the risk allows an organization to properly make an informed assessment on the value of investing in a strategy or project.

Opponents of OSS are concerned of the mere idea of freely available code, having someone to hold accountable in the case that something malfunctions, and the availability of support (O'Connor et al., n.d.). They argue that open access to the source code will allow malicious entities easier access and availability to compromising the code. Additionally, there is the issue of accountability: who will the DoN hold accountable for an open source issue, and how quickly can an essential community of volunteer code writers fix an OSS problem? Alternatively, as previously discussed, there are those who would argue that open availability of the source code will result in a myriad of users working together to perfect the code. Both the proponents and opponents of OSS seem to argue, primarily, on the security of OSS. As seen in Table 4, OSS generally has fewer defects than its competitors. However, organizations must still incur the burden of establishing and maintaining security, conducting audits, and correcting vulnerabilities. For a comprehensive understanding of OSS, enterprises must consider security and its role in OSS.

E. SUMMARY

Chapter III analyzed the different variables that must be considered when determining cost for OSS. Organizations must explore both direct and indirect cost as well as qualitative cost that are often intangible. When determining the cost, it is important to also consider the total cost of ownership over the life cycle of the software or the program. Once all foreseen costs are accounted for, it would then be beneficial to look at an analysis of alternatives to OSS. OSS has several cost advantages over proprietary software such as the software and license cost and the reduced external experts that are required to provide technical assistance to a program or software, but organizations must still look at their specific organizational strategy and capabilities to determine if OSS is the best option for them to pursue. This chapter also looked at vulnerabilities and case studies of OSS projects, proving that implementation of OSS in certain areas of an organization, has the potential to be secure and benefit large organizations in terms of cost savings. It is clear that the use of OSS can have significant cost savings for the DoN, if it is strategically planned and implemented.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. THE SCALED APPROACH AND RESEARCH FINDINGS

When this study began, the hypothesis was that if our research found that the DON could not go to full-scale open sourcing, this thesis would suggest a scaled approach. OMB has since directed that Federal Agencies make customized source code "broadly available for reuse across the Federal Government" (Scott & Rung, 2016, p. 1). Not only would this eliminate duplicative costs, it would improve collaboration between government and industry. The memo also developed a three-year pilot program requiring each agency to release 20 percent of their newly developed code as OSS for the duration of the program (Scott & Rung, 2016). In order to encourage collaboration with other agencies and industry, agencies be able to access the rights to the custom-developed code (Scott & Rung, 2016). Additional guidance for implementation would be provided on the code.gov website. There are exceptions to the policy for sharing of code that would jeopardize national security, personal security, or be against the law (Scott & Rung, 2016). Every software development effort must consider using OSS or COTS before developing custom code. To meet the restrictions of the OMB memo, a DON system will have to develop at least 20% of its new code as either OSS or reuse code unless it meets the exception in this policy. Given the potential limitations due to cyber security or legal requirements, a scaled approach proved to be prudent, as seen in the Beaumont hospital case. Beaumont chose to use OSS for administrative functions but not for some of their more technical software. In the Beaumont case, they wanted the ability to share information with other hospitals and had certain processes that they wanted to use as proprietary software. They assessed their organization and explored how to make OSS work for their situation (Fitzgerald & Kenny, 2004).

The findings from this research show that overall development and implementation of strategic goals will lay a foundation for further guidance and follow-on research. The bases for establishing strategic goals for OSS implementation were supported by the review and research of policies that specifically address the use of open sourcing. Based on the review and research of these policies, strategic goals for the implementation of OSS in the DoN should be addressed according to the topics of infrastructure, software development, security and life cycle cost, and reduction savings. To better assist with this process, the DoN should consider a new development methodology. The development and operations (DevOps) methodology may provide an efficient way for the DoN to implement OSS and applications. The use of DevOps and strategic goals can help advance the policies that were set forth in DoD correspondence to routinely consider OSS as a viable option for applicable programs.

A. STRATEGIC GOALS AND SUPPORT

Strategic goals will provide the "road map and support basis" to the integration of OSS in the DoN. These goals were first highlighted in the MITRE report and, as mentioned in Chapter II, OSS remains prevalent in this specific areas of the DoN. The researchers also believe that, given the current state of DoD policies pushing the use and exploration of OSS, these are the strategic goals and areas where the DoN can effectively progress the use of OSS. The strategic goals are as follows: infrastructure support, software development, security and life cycle cost and reduction savings.

1. Infrastructure Support

The first strategic goal is to ensure the current infrastructure support can maintain the use of OSS in its current state. According to the OSS & The DoD article, "On August 8, 2016, the White House Chief Information Officer (CIO) released a Federal Source Code Policy that calls for new software to be built, shared, and adapted using open source methods to capitalize on code that is secure, reliable, and effective in furthering our national objectives" (FitzGerald et al., 2016, p. 6). The policy requires that "new custom-developed source code developed specifically by or for the Federal Government to be made available for sharing and re-use across all Federal agencies ... [and] Federal agencies to release at least a portion of new custom-developed Federal source code to the public" (FitzGerald et al., 2016, p. 6).

According to the Memorandum, "Federal Source Code Policy: Achieving Efficiency, Transparency, and Innovation through Reusable and OSS," "This policy also establishes a pilot program that requires agencies, when commissioning new custom software, to release at least 20 percent of new custom-developed code as OSS for three years, and collect additional data concerning new custom software to inform metrics to gauge the performance of this pilot" (Scott & Rung, 2016, p. 2). This directly addressed the software improvement, efficiency, transparency, and cost savings provided by the use of OSS within the government (Scott & Rung, 2016). In order to achieve the desired effects of implementation, the CIO proposed a new program that would involve new programs to release 20% of its code for application and research (Scott & Rung, 2016). The release of the research data in 2019 will provide the metrics for data collection, research, and outcomes from the use of open sourcing within the DoD (Scott & Rung, 2016). The main point provided by the memo and research is that the DoD currently has an IT infrastructure that can maintain, support, and provide feedback for OSS. Although OSS has been part of DoD infrastructure for many years, it has now become a more viable option as costs rise and budgets are minimized within the Federal Government.

Currently, DoD Infrastructure has many applications and different software that comprise its architecture. MITRE reported, "Since much of the infrastructure of the Internet was created under the FOSS model, its infrastructure applications such as Apache are generally older, more functionally mature, and less likely to fail than much more recent proprietary equivalents" (MITRE Corp, 2003, p. 17). The MITRE report highlighted more than 65 OSS applications used in DoD infrastructure support the overall architecture that is the framework for the DoN infrastructure system (MITRE Corp, 2003). Therefore, the implementation of OSS within the DoN infrastructure is not necessarily a foreign idea. In fact, replacing these "already in use" OSS applications with proprietary applications will not necessarily show an improvement in infrastructure design, efficiency, or reduction in cost. OSS is so embedded and in perfect function with many software applications today that its complete removal from DoN applications would likely raise the financial cost (MITRE Corp, 2003).

The current IT infrastructure system is capable of handling OSS projects. The case study of the Coast Guard Machinery Control System (CGMCS), provides a good example of OSS improving the functionality, scalability, and cost effectiveness of the Machinery Control System (MCS) for both the Navy and the Coast Guard. The MCS allowed engineers to view various components such as electrical systems, propulsion systems, machinery, and other various hardware and software components through a Human Machine Interface (HMI) design (Reed, Cohen, Majumder, Chonko, & Walker, 2013). The case study explained that the current MCS continued to become more technologically complex and unable to become standardized amongst each ship. The technological complexity caused the life cycle cost to rise over the life of the MCS. In order to minimize rising cost, increase scalability, and standardization across all ships, the Coast Guard and Navy began implementing the new CGMCS on National Security Cutter (NSC) WMSL-750 Class Ships (Reed et al., 2013).

The goal of the CGMCS was to reduce development time and cost for individual ship classes. The CGMCS allows for the commonality of more affordable hardware and adaptable software that can be utilized in any ship (Reed et al., 2013). This program allowed for adaptable software that was used in different ship classes. The software used in this system ran on different vendor technologies such as Linux or Windows and uses open source libraries, as seen in the Figure 9.

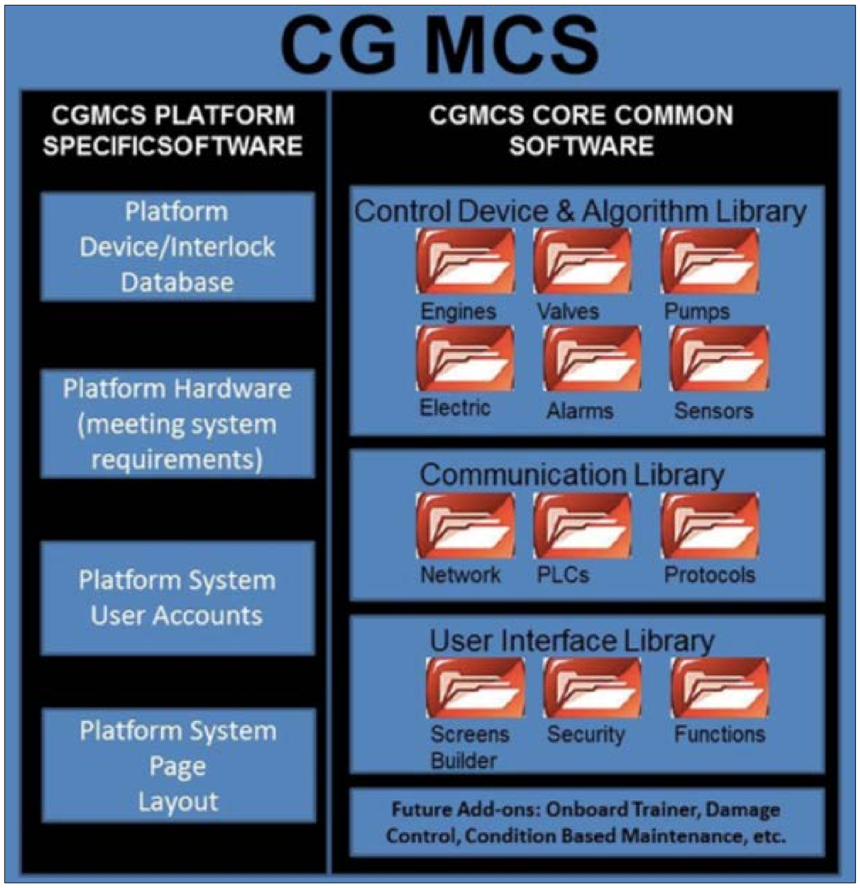

CG MCS
CGMCS PLATFORM SPECIFICSOFTWARE
CGMCS CORE COMMON SOFTWARE
Platform Device/Interlock Database
Platform Hardware (meeting system requirements)
Platform System User Accounts
Platform System Page Layout
Control Device & Algorithm Library
Engines
Valves
Pumps
Electric
Alarms
Sensors
Communication Library
Network
PLCs
Protocols
User Interface Library
Screens Builder
Security
Functions
Future Add-ons: Onboard Trainer, Damage Control, Condition Based Maintenance, etc.

Figure 9. CGMCS Common Software Functional Libraries. Source: Reed et al. (2013).

The scalability, implementation and functionality of this system proved the DoN has an infrastructure that can allow for further use of OSS. However, the CGMCS is a specific case in which the attributes of OSS work effectively for the need of the Navy. A scalable approach for OSS will prove to be beneficial for some applications since not all systems or hardware will use or benefit from OSS.

2. Software Development

The second strategic goal is to prioritize software development within the DoN and the DoD as a whole. Many new high-priority topics are of great concern today. Examples of new IT innovation include rapid new developing technologies, new airframe platforms, emerging cyber security threats, and the newly adopted information warfare offensive. Yet, in order to ensure there is functionality between each of these functions and their respective commanders, more emphasis needs to be directed at software development.

Software development should be the driving factor to ensure that each one of these priorities is able to perform its functions so that they operate smoothly. FitzGerald et al. stated, “Unfortunately, software development is not currently a high-profile, high-priority topic in the discussion about diminishing U.S. military technical superiority. It should be.” (FitzGerald et al., 2016, p. 5). The authors also discussed the issues related to much of the DoD’s technical problems with new technology. FitzGerald et al. argue that software development is not on the forefront of DoD priorities, yet much of the new technology created today, like weapon systems and aircraft technology, requires software that is developed, can be modified, and is interoperable; such as OSS (FitzGerald et al., 2016).

Much of the success of these new programs depends largely on the ability of the DoD to develop new software. OSS allows for the exact building of the needed software to fit the design in which it is applied to. The OSS methodology is built collaboratively. It is often built with various entities collaborating in a private or public manner (FitzGerald et al., 2016). The collaboration of many experts is what allows for OSS to steadily move forward to adapt to any modifications or design fixes that are required by the end user. In fact, the civilian private sector has prioritized software development in their private businesses and corporations.

The prioritization of software development has led the private industry to achieve success in the IT community as where the DoD insufficiently lags behind (FitzGerald et al., 2016). According to FitzGerald et al.,

> In recent years, the private sector has become increasingly reliant on OSS, which underpins critical software infrastructure from enterprise applications to smartphones and advances from artificial intelligence to electric cars. But while the commercial world has installed repeatable and scalable frameworks that improve the software it uses, the DoD struggles to keep pace. (FitzGerald et al., 2016, p. 4)

The success of prioritizing software development by private organizations such as google, Apache, Linux, Facebook, and Amazon can present a perfect example of how the

DoN and the DoD as a whole can optimize the best solutions for success in software development.

Integration of more OSS into the DoN may eventually lead to a push of more software development. This will allow software development practices to become much easier to prioritize for DoD officials (FitzGerald et al., 2016). In order to replicate IT success and efficiency in programs, like Google, Linux, and Apache, the DoN must advocate for IT to be considered a high priority. This emphasis on software development has the potential to greatly benefit the DoN in future technology performance, acquisitioning of software, interoperability, and the overall control of software development.

3. Security of OSS

The Strategic goal for the implementation of security in OSS is to provide understanding that security measures are already in place. A further setting of policies and procedures should be created to help provide understanding and alleviate security concerns of integrating more OSS into the DoN. The MITRE Corporation researched at least 44 organizations involved in DoD security use Free OSS (FOSS) (MITRE Corp, 2003, p. 20).

a. Open Source Security Concerns

Many in the government believe that an infrastructure that supports using OSS is at higher risk to cyber-attacks or more vulnerable to hackers, however this is not the case. According to Joseph, a technology writer for opensource.com,

> The open source projects that have open sourced their infrastructures have proven the value of allowing multiple companies and organizations to submit educated bug reports, and even patches and features, to their infrastructure. Suddenly you can invite part-time contributors. Your customers can derive confidence by knowing what your infrastructure looks like under the hood. (Joseph, 2018, p. 1)

Infrastructure, integrity, and security is not compromised by OSS. In fact, the software provides for more opportunities to allow the de-bugging of programs, and the increase of safety from cyber-attacks (Joseph, 2018). The OSS security analysis is done by different organizations and programmers because they are given access to the source code which allows these programmers, which are contracted by the DoD, to find and help prevent cyber incidents from occurring. In others words, there is more visibility and eyes on the software in order to determine if the software has been compromised in anyway. OSS provides the visibility that propriety or closed software cannot provide because it is only being analyzed by one entity; if it is analyzed at all.

When considering security concerns of OSS applications, many believe that proprietary software is considered “more” safe than OSS, however the issue much more complicated than that. Dunn and Wheeler interviewed several government employees on their cyber security expertise on OSS. According to the authors,

> There is a concern over the ease of getting malware into OSS. Actually, it’s pretty easy to get malware into proprietary software too. OSS is unique in that it gives complete visibility into the supply chain. Another government employee said, “Just because you cannot [review] the source [of proprietary software] does not mean the software is safe ... I would rather know where it came from so I know what to target in my evaluation. (Dunn & Wheeler, 2013, p. 11)

The article by Dunn and Wheeler addressed many security concerns from government officials. One main concern from the government was that OSS was “easier” for another individual to slip in malware or that the software in general was not safe. The advice given by the experts in the article address that OSS actually provides a safer option because of the visibility that it provides. The visibility allows programmers to detect malware, viruses, or any other foreign entity within the code. Therefore, providing solutions to quickly address and eradicate any issues.

Another reason that government officials advocate proprietary software over OSS is that proprietary software is always considered the more “safe” and consistent solution to software in the DoD. However, based on research and expertise, Dunn and Wheeler advocate that just because software is proprietary does not make it safe from cyber-attacks. In fact, an issue that arises with proprietary software is that the software code cannot be viewed, therefore allowing only the producer of the software to view all security issues within the software. This essentially leaves all security concerns of that software to be addressed by the producing entity's inspectors; in theory, decreasing the chances of discovery of security issues within the software code (Dunn and Wheeler, 2013).

Despite the praise that OSS receives for the large community of contributors and security checks on the software, there is evidence that this does not apply to all OSS supported components. The Snyk report raised serious security concerns for the general open source community. Snyk conducted interviews of open source maintainers and determined that most in the community have not had a security audit (Snyk, 2017). Figure 10 shows Snyk's results for OSS audits.

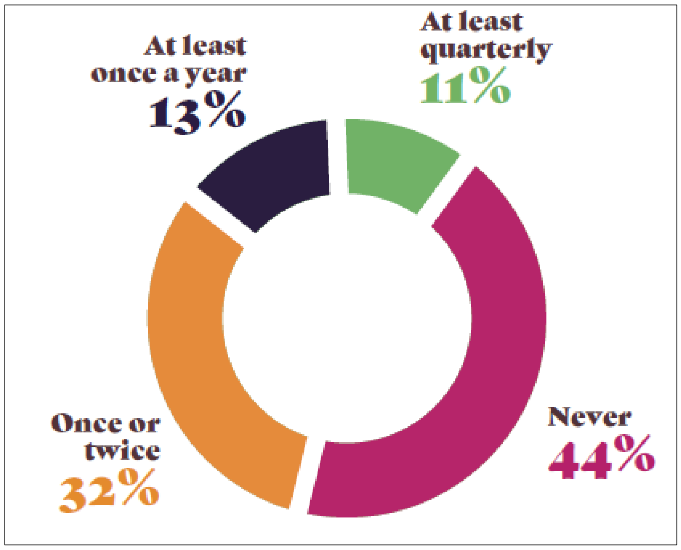


Figure 10. How Often Maintainers Audit their Code.
Source: Snyk (2017).

Audits are conducted to assist with finding vulnerabilities. This is an area that will have to be monitored and regulated if implemented in the DoN. In order for OSS to function with the minimal amount of risk, the DoN will have to conduct audits and apply security standards to each OSS program.

b. Security Controls

The National Institute of Standards and Technology (NIST) produces a series of documents designed to outline the responsibilities required by the Federal Information Security Management Act (FISMA) (NIST, 2013). The NIST SP 800–53 Rev 4 describes the security requirements, privacy controls, as well as the processes that are used to select security controls on IT systems (NIST, 2013). This document is important because as the use of OSS becomes more prevalent, there will be a continued need to monitor and assess the security and vulnerabilities of the IT infrastructure. The NIST 800–53 is only one of the many publications that assist with the security of programs, but it provides an example of the types of security controls and risk assessment that would be necessary in determining whether or not to use OSS in programs. The security controls that are found in the NIST 800–53 are "designed to facilitate compliance with applicable federal laws, Executive Orders, directives, policies, regulations, standards, and guidance" (NIST, 2013, p. x). In addition to meeting published guidelines and regulations, the security controls are also designed to protect information systems from a variety of threats in different scenarios and operating environments (NIST, 2013).

The NIST SP 800–53 provides the security controls; however, it is the organization's responsibility to assess the system operating risk, select the security controls, oversee implementation, monitor the functionality, and make adjustments to the system or software. This can be an extensive process that requires strategic planning and organizational commitment; the NIST SP 800–53 summarized best practices to achieve secure systems, as seen below:

- Clearly articulated security requirements and security specifications
- Well-designed and well-built IT products based on state-of-the-practice hardware, firmware, and software development processes
- Sound systems/security engineering principles and practices to effectively integrate IT products into organizational information systems
- Sound security practices that are well documented and seamlessly integrated into the training requirements and daily routines of organizational personnel with security responsibilities

- Continuous monitoring of organizations and information systems to determine the ongoing effectiveness of deployed security controls, changes in information systems and environments of operation, and compliance with legislation, directives, policies, and standards
- Information security planning and system development life cycle management. (NIST, 2013, p. 5)

Before the DoN can select the security controls required for whatever OSS platform it intends to incorporate into the IT infrastructure or programs of record, leaders at all levels must consider the risk. As previously discussed, there are risks present in every software decision and OSS is not an impenetrable fortress, so risks will have to be considered before implementation. The NIST 800–53 created a diagram that considers risks and establishes communication about those risks at all levels of management. This diagram is found in Figure 11.

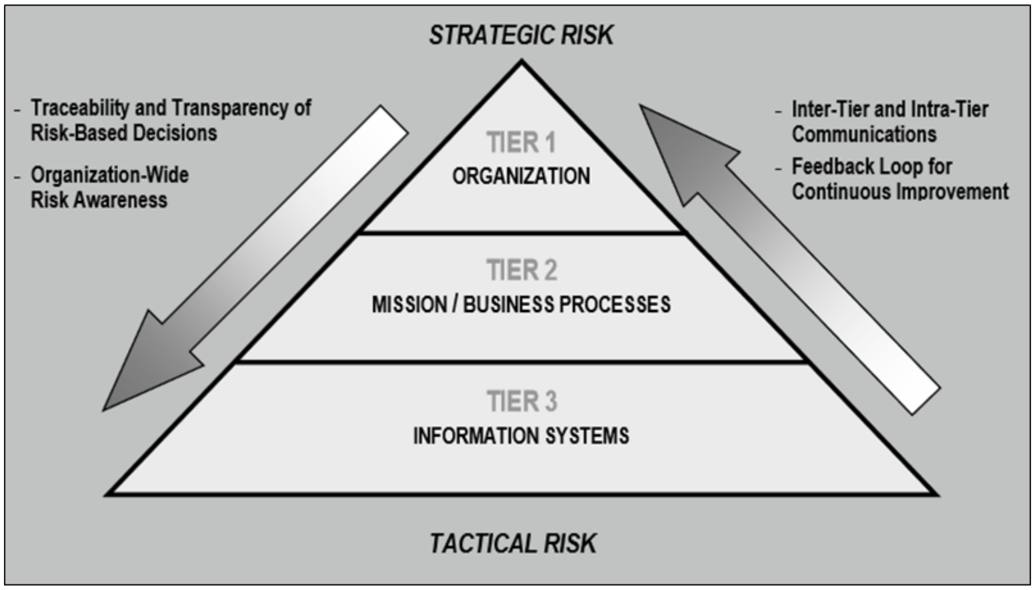

STRATEGIC RISK
- Traceability and Transparency of Risk-Based Decisions
- Organization-Wide Risk Awareness
TIER 1
ORGANIZATION
- Inter-Tier and Intra-Tier Communications
- Feedback Loop for Continuous Improvement
TIER 2
MISSION / BUSINESS PROCESSES
TIER 3
INFORMATION SYSTEMS
TACTICAL RISK

Figure 11. Three Tiered Risk Management Approach.
Source: NIST (2010).

Figure 11 shows how risk span across management tiers and how the feedback loop will affect both tactical and strategic plans. It will be important for the DoN to carefully consider risk at all tier levels when picking the appropriate security controls for every system or software it uses. Each tier level has different goals from Tier 1, which priorities the overarching mission and drives decisions, strategy, and investment strategy to Tier 2, which defines processes to support functions, determines security categories, and establishes the enterprise architecture (NIST, 2013). Tier 3 utilizes the Risk Management Framework to incorporate information systems (NIST, 2013).

The Risk Management Framework (RMF) is a six-step process that is used to address security concerns of an organization like the DoN. The NIST 800–37, Rev 1 depicts the RMF, as seen in Figure 12.

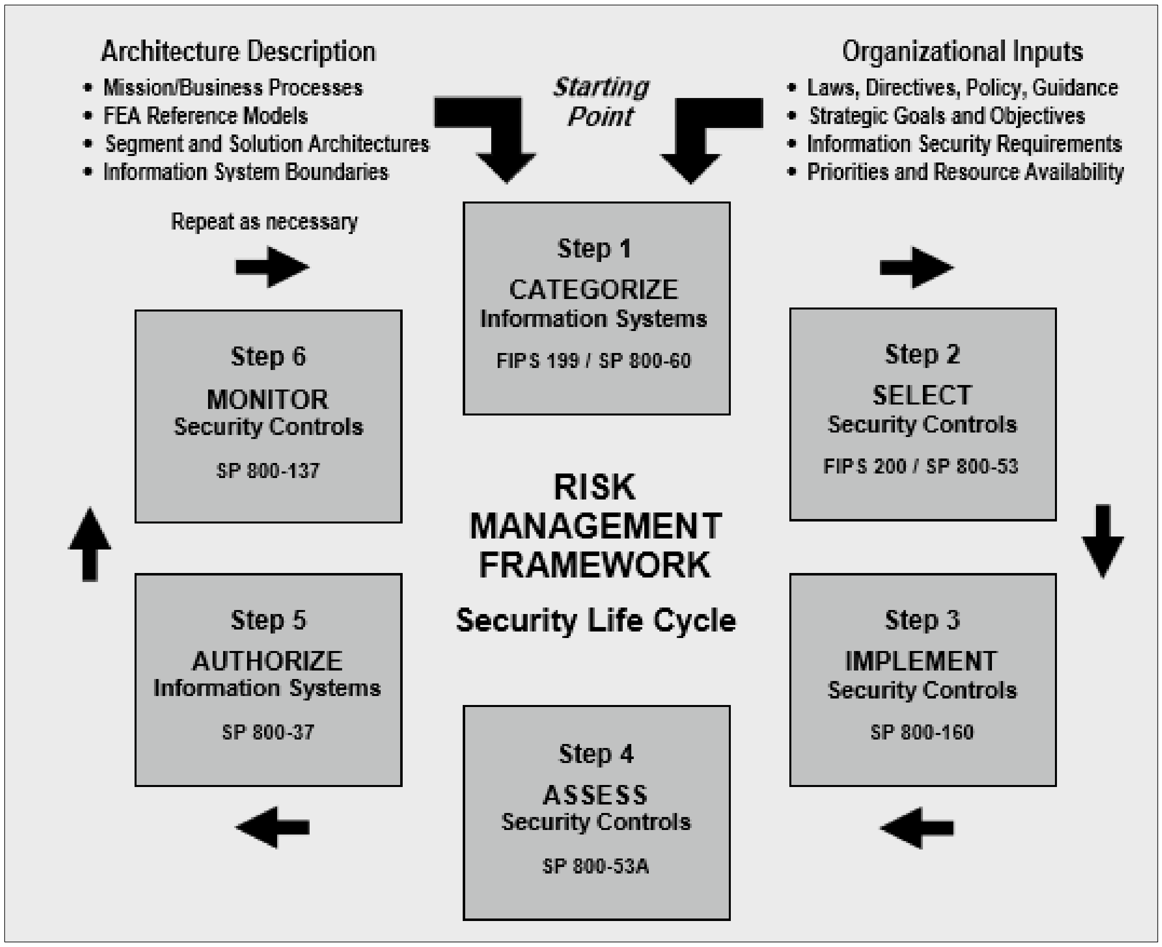
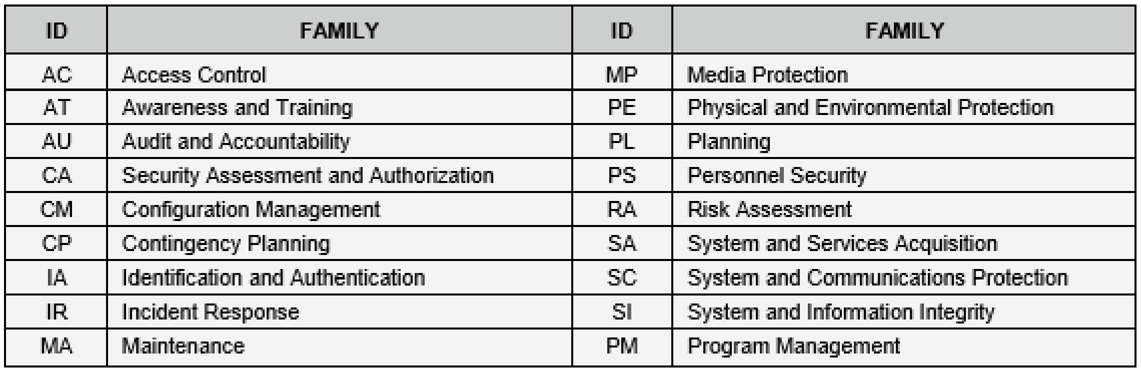

Architecture Description
Mission/Business Processes
FEA Reference Models
Segment and Solution Architectures
Information System Boundaries
Starting Point
Organizational Inputs
Laws, Directives, Policy, Guidance
Strategic Goals and Objectives
Information Security Requirements
Priorities and Resource Availability
Repeat as necessary
Step 1 CATEGORIZE Information Systems FIPS 199 / SP 800-60
Step 2 SELECT Security Controls FIPS 200 / SP 800-53
Step 3 IMPLEMENT Security Controls SP 800-160
Step 4 ASSESS Security Controls SP 800-53A
Step 5 AUTHORIZE Information Systems SP 800-37
Step 6 MONITOR Security Controls SP 800-137
RISK MANAGEMENT FRAMEWORK
Security Life Cycle

Figure 12. Risk Management Framework. Source: NIST (2010).

Each step has a significant purpose designed to supplement the entire risk assessment and program integration process, but the step that deals mostly with implementing OSS across all systems in the DoN is Step 2: Select Security Controls.

The security controls listed in the NIST 800–53 are organized into families. The 18 families cover a large range from Access Controls, Awareness and Training, to Media Protection, Incident Protection, and Program Management. These families cover many aspects of oversight, individual action, supervision, automated mechanisms and many more (NIST, 2013). As OSS becomes used more often, the DoN will have to view each individual system or program, in the same manner it currently does, which is in accordance with the governing regulations. The reason that this paper targets the NIST 800–53 is to highlight that is already policy in place to govern OSS and to list specific areas of concern; in terms of OSS. Because OSS is rapidly expanding on the civilian side of IT, it seems only natural that the DoN will have to further OSS in order to maintain compatibility, flexibility, and competiveness with foreign military organizations. As open source expands to work in all areas of the DoN, OSS will need a wide variety of security controls to address IT capabilities. The family of security controls highlighted in the NIST 800–53 are located in Table 9.

Table 9. Security Control Identifiers and Family Names.
Source: NIST (2013).



ID	FAMILY	ID	FAMILY
AC	Access Control	MP	Media Protection
AT	Awareness and Training	PE	Physical and Environmental Protection
AU	Audit and Accountability	PL	Planning
CA	Security Assessment and Authorization	PS	Personnel Security
CM	Configuration Management	RA	Risk Assessment
CP	Contingency Planning	SA	System and Services Acquisition
IA	Identification and Authentication	SC	System and Communications Protection
IR	Incident Response	SI	System and Information Integrity
MA	Maintenance	PM	Program Management

Each family has sections that include guidance on control, supplemental guidance, control enhancements, and references (NIST, 2013). The NIST SP 800–53 goes on to describe common controls and how they can be positively employed and adversely affected. Ultimately, the DoN will need to explore publications such as the NIST 800–53 for integration of OSS. Fortunately, publications like this are already in existence and already have specific guidance and examples of OSS integration.

As the security concerns of an integrated OSS network increase with greater integration within the DoN, leaders can be assured because of the evolution of its security measures and ever-growing security features that programmers will be able to openly share concerns, create patches, and find solutions for all DoD software and security related issues. OSS is an authorized option for both secure and non-secure programs. Security conditions must apply; however, OSS are feasible and viable.

4. Life Cycle Cost Reductions/ Savings

The final strategic goal of OSS is to analyze the cost savings of implementing OSS into the DoD; more specifically the DoN. First, the current structure of proprietary computer government contract business allows little room for competition for large IT solutions (MITRE Corp, 2003). Generally, industry powerhouses consume the DoN IT market. The lack of competition prohibits the ability of the DoN to save money or allocate resources for other necessary projects.

As in any business-related transaction, the consumer prefers to have options. These options then allow each producer to compete for business; this creates better products at lower prices. When OSS is concerned, “Without the constant pressure of low-cost, high-quality OSS product competing with the closed-source products, the closed-source vendors could more easily fall into a cycle in which their support costs balloon and costs are passed on to their locked-in customers” (MITRE Corp, 2003, p. 22). This is a small example of the costly repercussions of not using OSS.

Furthermore, longevity in OSS is where the true lower cost will be presented, according to Use of Free and Open-Source Software (FOSS) in the U. S. DoD, “Costs would drop in both the short and long term as costlier applications are replaced by FOSS products such as Apache that are almost universally considered to be higher quality” (MITRE Corp, 2003, p. 23). Over time, the strategic adoption of OSS in certain areas of the DoN could benefit the service as it is able to leverage a combination of OSS and proprietary systems to increase efficiency and decrease costs.

B. DEVOPS

As OSS continues its growth and expansion within military software systems, it is hard to deny that the methodology to approaching this type of software is not going to change. No longer will proprietary software always be applicable to the ever-changing environment of IT. As the expansion of OSS continues, the DoN must learn to adopt to new IT practices in order to become faster, agile, and increase capacity to innovate. The authors of the article "What is DevOps," explain the meaning of DevOPs as:

> DevOps which is short for Development and Operations is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. (Amazon Web Services, 2018, p. 1)

The cultural embrace of DevOps is what allows for the mixing of development and operations to provide speed of delivery, agility and user proficiency for organizational success in software development and delivery. DevOps and OSS form a synergistic relationship in which each entity helps the other become successful. OSS provides the flexibility, cost-savings, and avoids vendor lock-in, while DevOps provides rapid deployment, improved collaboration between development teams, and manageable infrastructure (Lyman, 2018).

1. How DevOps Works

When it comes to allowing DevOps to integrate and essentially change the software climate of an organization, both DevOps and OSS have to synchronize in harmony. In the article, "Open Source leads to DevOps Success," the author Jay Lyman states, "According to enterprise DevOps users have also routinely reported that the modularity and componentization of open-source software is a good fit for DevOps, which tends to involve a broader array of tools and technologies" (Lyman, 2018, p. 1). The typical DevOps model does not have separate teams as most conventional or proprietary software is developed. DevOps instead combines development and operations, according to the article "What is DevOps?,"

> Under a DevOps model, development and operations teams are no longer "siloed." Sometimes, these two teams are merged into a single team where the engineers work across the entire application life cycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function. (Amazon Web Services, 2018, p. 1)

DevOps uses a synergistic approach to combining aspects of development and operations to form a faster produced software. The DevOps model is designed to build software by building, testing, and releasing once the software has been developed by its engineers. The feedback loop from the customers provides the monitoring of the software and provide plans to the providing company for planning future updates and new types of software. An example of this DevOps approach by Amazon Web Services is provided in Figure 13.

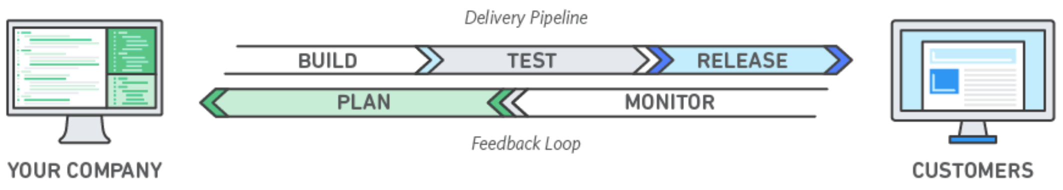


Figure 13. DevOps Concept and Information Flows Model.
Source: Amazon Web Services (2018).

As IT capabilities continue to move at a fast pace with innovation, emerging markets such as big data, social media, cloud computing, and mobile applications are recognizing the benefits and business value of DevOps. The business benefits the DoN can receive from DevOps is the maximization of speed of its delivery of a product or service, from initial development to production to customer feedback to enhancements all based on feedback (Coyne & Sharma, 2017, p. 4). Listed below are benefits that are provided by DevOps:

- Speed: Move at high velocity so you can innovate for customers faster, adapt to changing markets better, and grow more efficient at driving business results.
- Enhanced Customer experience: Providing differentiated and engaging customer service which builds customer loyalty and positive business relations.
- Rapid Delivery: Increase the frequency and pace of releases so you can innovate and improve your product faster. The quicker you can release new features and fix bugs, the faster you can respond to your customers' needs and build competitive advantage.
- Increased Capacity to Innovate: In order to increase capacity to innovate, DevOps focuses on a lean thinking approach. In this approach the goals are to reduce waste, rework, and to shift resources to higher value projects.
- Reliability: Ensure the quality of application updates and infrastructure changes so you can reliably deliver at a more rapid pace while maintaining a positive experience for end users.
- Scale: Operate and manage your infrastructure and development processes at scale. Automation and consistency help you manage complex or changing systems efficiently and with reduced risk.
- Improved Collaboration: Build more effective teams under a DevOps cultural model, which emphasizes values such as ownership and accountability.
- Security: Move quickly while retaining control and preserving compliance. You can adopt a DevOps model without sacrificing security by using automated compliance policies, fine-grained controls, and configuration management techniques. (Amazon Web Services, 2018, p. 1)

By employing each benefit listed, this allows the encompassing experience of DevOps to operate fluidly and provide the best experience possible when it comes to IT and development. Author Mano Paul states, "DevOps doesn't seek to erase the differences between the two disciplines of software development and IT operations, but instead builds a bridge to make them work better together while continuing to follow traditional processes in each discipline independently" (Paul, 2014, p. 1). Overall, it is the ability of DevOps to work synergistically that allows for a fluid IT operating system. This is why OSS is such an important asset when it comes to DevOps. OSS allows for software systems to talk to each other and help increase interoperability which is one main benefit of DevOps.

2. DevOps Culture

The DoD is being left behind in the IT realm because of strict cultural norms and the bureaucracy of acquiring new services and goods. In order to stay relevant within the IT world and up-to-date with competing organizations, the cultural mindset of DevOps should be embraced. This would allow for more efficient systems, more fault tolerant systems, more routine deployments, more collaboration, continuous monitoring, and interoperable software systems. Figure 14 shows the many aspects of DevOps culture.

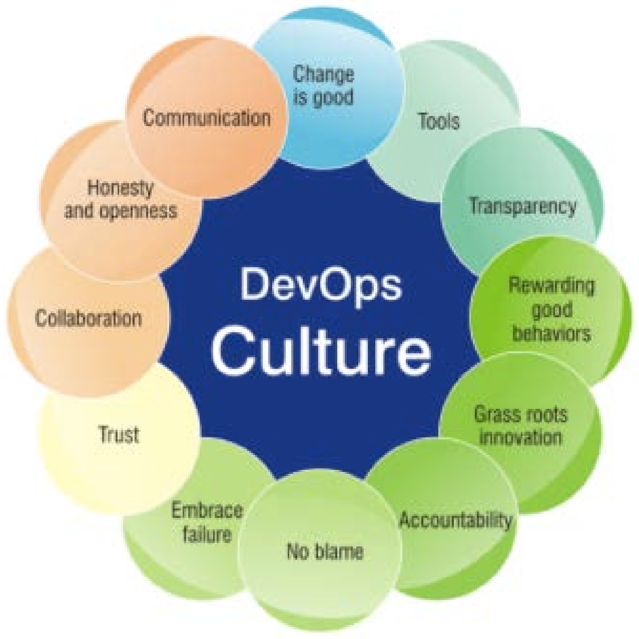


Figure 14. DevOps Culture Loop. Source: WebSenor (2018).

Regarding DevOps, Endsley stated, the "first steps into DevOps are about examining your culture and practices, identifying the barriers to cross-team communication and coordination, and taking the steps necessary to bridge communication between your development and operations teams" (Endsley, 2018, p. 1). Cultural mind-shift and embracing DevOps allows for the agility and continuous delivery of improved software life cycles. The integration with OSS is a symbiosis of software and operations allowing for the best experience between software developers and the users of the software to create the best product needed to accomplish any mission.

C. SUMMARY

Chapter IV provided a scaled approach to determine the feasibility of OSS implementation in the DoN. The implementation is presented in four strategic goals, followed by a brief summary of DevOps in order to present the type of environment/culture OSS demands in order to thrive.

The first strategic goal of infrastructure provided the understanding that the current infrastructure can support OSS. Many OSS programs today operate under the current IT infrastructure, therefore before OSS integration can begin additional policies and research may help ease the transition from closed-source systems to open source systems. The second strategic policy of software development explained how the DoN can gain an advantage by prioritizing software development much like its civilian counterparts. Prioritization provides the ability to create software that can meet demands of technology.

The third security strategic goal discussed security measures, security issues, security vulnerabilities, and provided in depth analysis of the NIST SP 800–53 and security protocols. The analysis provided an understanding of OSS security measures and its ability to provide overall protection to the user. Cost benefits of OSS were reviewed and establish in the fourth strategic goal. OSS proved to have advantages over proprietary software in its ability to be cost effective.

Lastly, DevOps although not fully reliant on OSS, proved to be beneficial in its ability to provide a culture that is conducive to operating on OSS. Likewise, OSS although not reliant on DevOps, is complimentary to DevOps and is greatly benefited by its presence. Chapter IV presented a scaled approach of strategic goals along with the proper environment supported by DevOps, to create an operational environment beneficial to implementing OSS in the DoN.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION AND FUTURE WORK

A. THESIS SUMMARY

The scalable adoption of OSS into certain areas of DoN is a critical step toward evolving the IT capabilities of the service. As the IT realm continues to rapidly expand, the DoN will have to find cost effective methods to stay on pace with both their adversaries and civilian counterparts; who also have vested interests and are intertwined in DoN programs. This research clearly defined OSS and addressed the DoN policy that allows the use of open software. The research also concluded that OSS is becoming more common in business and industry practices. With a large number of organizations undertaking open source solutions, the use of OSS in the DoN can help realize the need to keep up with technology, while minimalizing the reliance of expensive proprietary software and long-lasting contracts. The ability to see the code will help internal service members learn from some of the industry's best software engineers. Furthermore, the research shows that OSS is generally cost effective, especially in terms of licensing. However, long-term support for OSS can be expensive, depending on the program and organization. When deciding what type of software to use on a project, and organization must look at both direct and indirect cost. In addition to the cost savings, OSS offers many intangible qualities that must be accounted for and factored into the total cost of ownership. Researchers have proved that OSS can still be reliable and high quality products.

OSS is compiled by a large community of developers which provides variety and more scrutiny of the code. This community is authorized to freely contribute to OSS but must adhere to OSS licenses. This includes all rules that pertain to distribution and version control. It is the organization's responsibility to decide which OSS and version would best suit their business strategy and goals; especially since, as seen in the various case studies, OSS is not always the answer. Opponents of OSS are wary of the availability of the source code and often tout the practice of security through obscurity. This is a valid concern, but proponents will argue that the large community provides OSS with layers of defense-in-depth. Both of these general perceptions have merit, but the research shows that OSS has fewer defects than CSS. Ultimately, all software has vulnerabilities. It is up to the operating organization to ensure that it conducts a proper risk assessment and monitors all software and programs for security compliance. Different methodologies are available and are encouraged to foster and enhance OSS. DevOps is a methodology that is proven to work with OSS. It provides the ability to continually monitor and deploy software and products, which increased speed and durability in the IT infrastructure.

B. RESEARCH QUESTIONS

This research sought to answer several questions regarding OSS and the feasibility of use within the DoN. The research included a review of DoD and DoN policies, case studies, and articles published by topic experts. After concluding the research, the authors determined the following conclusions.

1. What are the Processes, Criteria, and Responsibilities for Publishing Publicly Releasable Software?

If the DoN chooses to use OSS for projects and programs of record, they will have to adhere to OSS licenses and the specific rules that apply to each specific license. The various licenses have different rules that describe how to use the specific software. These rules provide instructions on managing OSS version control and public distribution. Program managers and software developers will have to meticulously ensure that they are operating and using the software as prescribed by the pertaining license. Additionally, the DoN can use websites described in this research, such as forge.mil and code.mil, to facilitate the use and public release of OSS.

2. What is the Feasibility of using OSS for DoN Programs, from Unclassified through Classified?

As highlighted in Chapters II and III, the DoD already uses OSS in certain areas and programs. The MITRE report highlighted specific areas that OSS is prevalent and these areas would be the logical place to monitor and pursue further development of OSS within the DoN. Forge.mil reported OSS programs on both the unclassified and classified networks, so the research has determined that the "feasibility" is already a reality. Future research should explore classified programs of record and how, if at all, OSS can be implemented into such programs.

3. Have there Been any Programs in the Federal Government, DoD, DoN that have Incorporated OSS?

The research found the DoN already has programs implemented in various places throughout the organization. A major program is the VistA program, which is an OSS DoD Veterans Affairs program that in now used by several countries all over the world. The 2003 MITRE report found that 115 open source applications were found in 251 instances. Policies such as the 2009 DoD CIO memorandum and the 2016 OMB open source memorandum encourage the DoD to continue to explore new ways of implementing OSS for greater information sharing, transparency, cost savings and reduction of work.

4. Was there a Discernable Cost Benefit for the Program? How would/could OSS Affect Maintenance Cost?

The main cost benefits to OSS are the ability to save on software purchase price and the savings incurred on obtaining proprietary licenses. The research addressed that the software costs are approximately 8% of the total cost of a program. With this in mind, other aspects must be considered when addressing cost savings. When the DoN considers where and how it should use OSS, it will have to carefully consider all aspects for every potential OSS program or application. Case studies were presented that showed cost benefits for both open source and proprietary options. In order to obtain significant cost benefits using OSS, the DoN must consider each OSS use and also the product support and potential maintenance for that software or program. The research shows that it is possible to significantly save using OSS; however, there are a myriad of considerations and planning factors that must occur in order to realize a cost saving goal.

5. What Are the Cybersecurity Requirements or Challenges for Implementing OSS?

OSS must adhere to the same cybersecurity requirements as all other software used in the DoN. There are procedures and controls that are set forth in current regulations and policies. The difference with OSS, is that it will require an entity to ensure that the software is assessed for vulnerabilities and audited ensure regulations are being followed. OSS, generally, has a large community of developers scrutinizing the software and companies such as Coverity providing the means to scan for vulnerabilities. Even though there is research showing that OSS can be secure, it will still require oversight and someone to hold accountable for unplanned work, maintenance, and defects.

6. If it Is Not Possible to Obtain All of the DoN's Requirements using OSS, would a Scaled Approach be Cost Effective?

Large technology companies are quickly moving and encouraging OSS. Even if the DoN does not pursue its own development of software using OSS, it will eventually use OSS components as a part of a proprietary solution. In the aggregate, it may make sense to develop certain programs using only proprietary software. Thus, a scaled approach to OSS will allow the DoN to learn to use and perfect the integration and use of OSS.

C. FUTURE WORK

OSS is still not widely accepted enough to overcome issues of integration within DoN IT systems. This research revealed the issues and challenges that may rise when trying to implement a change as significant as OSS. Several examples, such as culture change from proprietary software, security perception, security implementation, and DevOps integration were seen as common themes throughout the research process. The research addressed these subjects; however, the issues listed are recommended for further research, implementation, and possible solutions.

1. Culture Change

In the DoD, "open source" is considered a risky venture. When trying to propose solutions to current proprietary programs, OSS must be considered a feasible option. Although many government applications already contain OSS, one would think that the culture shift from proprietary software to OSS would be easy. However, the amount of comfort that is provided by proprietary software is enough that government acquisition experts are willing to pay the extra expenses for the sake of having all their software needs contained in one package. As convenient as this sounds, it may add up to higher cost, unsatisfactory customer service, and vendor lock-in. OSS may provide a needed solution to the problem. However, in order to bring the realization of OSS into fruition, entire paradigm shifts in education, training, change, and implementation tactics will need to be addressed. Perhaps changes such as updating the Defense Acquisitions University curriculum would propel the DoD towards more routine OSS consideration. More work and additional research will need to be accomplished on this topic in order to further the culture change in the government.

2. DevOps

An issue discovered in the research was the option to use DevOps. As mentioned earlier, DevOps can be used in DoN in order to be able to fully compliment the possibilities of using OSS and the possibility of creating software that is designed for our DoN projects. Although OSS can stand alone without DevOps, research suggests that the reciprocity between DevOps and OSS allows for the maximization of OSS use. Future research should be conducted into DevOps and the pros and cons that the DevOps culture may present to the DoN.

3. Security

The security perception of OSS is that it always has risks that are attached to the software. In general, OSS is going to have users, internal or external to the organization, able to view, manipulate, and modify source code of government projects and development. Each organization must monitor controls and access to their specific systems and ensure that continued evaluation occurs. A popular concern is that OSS producers, developers, or contributors may intentionally hide malicious in the software. However, the research has concluded that proprietary software poses similar or more likely risk of the occurrence of malicious activity. This is due to the closed nature of the proprietary software. Proprietary software may also contain the exact issues as with OSS, therefore making it just as risky as OSS. One option to mitigate security risk, would be to allow the DoD to write more code, rather than simply contracting everything out. Doing so could provide more internal knowledge and oversight of the software that the DoN uses. Future organization-specific research is needed in the realm of OSS and security to explain, provide examples of security flaws for OSS and proprietary software, and to provide security recommendations of the overall operation of OSS in the DoN.

LIST OF REFERENCES

ActiveState Software. (2016). The true cost of OSS: Uncovering hidden cost and maximizing ROI. Retrieved from https://www.activestate.com/sites /default/files/pdfwp/whitepaper-true-cost-opensource.pdf

Almeida, D. A., Murphy, G., Wilson, G., & Hoye, M. (2018, Apr 27). Investigating whether and how software developers understand OSS licensing. Retrieved from https://doi.org/10.1007/s10664-018-9614-9

Amazon Web Services. (2018). What is DevOps? Retrieved from https://aws.amazon.com/devops/what-is-devops/

Asay, M. (2016). Bill Gates gets real about free software. Retrieved from https://www.infoworld.com/ article/3042247/open-source-tools/bill-gates-gets-real-about-free-software.html

Bromhead, B. (2017, Aug 17). 10 Advantages of open source for the enterprise. Retrieved from https://opensource.com/article/17/8/enterprise-open-source-advantages

Brown, G. (2011, Jul 8). Standards 101: Understanding the importance of open standards. Retrieved from https://blogs.technet.microsoft.com/openness/ 2011/07/08/standards-101-understanding-the-importance-of-open-standards/

Carey, R. (2007, Jun 5). *Department of the Navy OSS guidance* [Memorandum]. Washington, DC.: Department of Defense. Retrieved from http://www.doncio.navy.mil/ contentview.aspx?id=312

Center for Strategic and International Studies. (2007). Global policies on OSS. Retrieved from http://www.csis.org/component/option,com_csis_pubs /task,view/id,4009/type,1/

Chrzanowska, N. (2017, Mar 30). 17 most helpful node.js open source projects according to experts. Retrieved from https://www.netguru.co/blog/nodejs-helpful-open-source-projects

Coverity. (2014). Coverity scan: 2013 open source report. Retrieved from http://softwareintegrity.coverity.com/rs/appsec/images/2013-Coverity-Scan-Report.pdf

Coyne, B., & Sharma, S. (2017). *DevOps for dummies*. Hoboken, NJ: John Wiley & Sons, Inc.

Department of Defense. (2017, Feb 23). DoD announce the launch of code.mil: an experiment in open source. Retrieved from https://www.defense.gov/News/News-Releases/News-Release-View/Article/1092364/DoD-announces-the-launch-of-codemil-an-experiment-in-open-source/

Department of Defense Chief Information Officer. (n.d.). OSS frequently asked questions. Retrieved from https://DoDcio.defense.gov/Open-Source-Software-FAQ/

Dunn, T., & Wheeler D. A. (2013, Aug 29). OSS in government: challenges and opportunities. Retrieved from https://www.dhs.gov/sites/ default/files/ publications /Open%20Source%20Software %20in %20Government%20– %20Challenges%20and%20Opportunities_Final.pdf

Endsley, R. (2018) What is open source? Retrieved from https://opensource .com/resources/what-open-source

Fitzgerald, B., & Kenny, T. (2004, Aug 9). *Developing an information systems infrastructure with OSS*. IEEE Software. January-February 2004.

FitzGerald, B., Parziale, J., & Levin, P. L. (2016, Aug 30). Open source software & the Department of Defense. Retrieved from https://www.cnas.org/ publications/reports/open-source-software-and-the-department-of-defense

Free Software Foundation (2018). Free software is software that gives you the user the freedom to share, study and modify it: We call this free software because the user is free. Retrieved from: https://www.fsf.org/about/what-is-free-software

Gates, B., & Balmer, S. (2005). Shareholder letter. Microsoft Corporation Annual Report 2005. Retrieved from: https://www.microsoft.com/ investor/reports /ar05/staticversion/10k_sl_eng.html

Garrison, J. (2010, Oct 26). What is the Linux kernel and what does it do? *How to Geek.* Retrieved from: https://www.howtogeek.com/howto/31632/what-is-the-linux-kernel-and-what-does-it-do/

Heath, N. (2014). Ditching windows for Linux led to “major difficulties” says open source champion Munich. Retrieved from https://www.techrepublic.com/article/ditching-windows-for-linux-led-to-major-difficulties-say-open-source-champioon-munich/

Heintzman, D. (2003, Jul 5). An introduction to open computing, open standards, and open source. Retrieved from https://www.ibm.com/developerworks/ rational/library/1303.html

International Data Corporation. (2013, Jul). IDC customer spotlight: Driving effective community software collaboration at the DoD with forge.mil. Framingham, MA: IDC Go-to-market services. Retrieved from http://www.forge.mil/downloads/ casestudy_IDC_Forge.Mil.pdf

Joseph, E. K. (2018, Aug 23). Why open source should be the first choice for cloud native environments? Retrieved from https://opensource.com/ article/17/8/open-sourcing-infrastructure

Koltun, P. (2011, Dec). Free and OSS use: Benefits and compliance obligations: CrossTalk. Retrieved from http://www.scopus.com/inward/record.url? eid=2-s2.0-81555199735&partnerID=tZOtx3y1

Korolov, M. (2018, Apr 2) OSS security challenges persist. Retrieved from: https://www.csoonline.com/article/3157377/application-development/ open-source-software-security-challenges-persist.html

Llaguno, M. (2017, Oct 31). 2017 Coverity scan report: OSS—The road ahead. Retrieved from https://www.synopsys.com/ content/dam/synopsys/sig-assets/reports/SCAN-Report-2017.pdf

Lyman, J. (2018). Open source leads to DevOps success. Retrieved from https://techbeacon.com/open-source-leads-devops-success

Martin, G., & Lippold, A. (2011). Forge.mil: A case study for utilizing open source methodologies inside of government. Retrieved from https://link.springer.com/content/pdf/10.1007%2F978-3-642-24418-6_28.pdf

McMillian, R. (2012, Mar 20). Linus Torvalds: The king of geeks (and dad of 3). Retrieved from: https://www.wired.com/2012/03/mr-linux//

MITRE Corp. (2001). *A business case study of open-source software*. (MITRE Report No. MP 01B0000048). Bedford, MA: Retrieved from https://www.mitre.org/sites/default/files/pdf/kenwood_software.pdf

MITRE Corp. (2003). *Use of free and open-source software (FOSS) in the U.S. Department of Defense.* Version 1.2.0.4. (MITRE Report No. MP 02 W0000101). McLean, VA: Author. Retrieved from http://www.terrybollinger.com/DoDfoss/ DoDfoss_pdf_hyperlinked.pdf

Moore, J. M., & Stanton, A. (2018). Tiebreaks and diversity: Isolating effects in Lexicase Selection. Retrieved from https://www.mitpressjournals.org /doi/pdf/10.1162/isal_a_00109

National Institute of Standards and Technology Special Publication 800–37. (2010). Guide for applying risk management framework to federal information systems: a security life cycle approach. Retrieved from https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-37r1.pdf

National Institute of Standards and Technology Special Publication 800–53. (2013). Security and privacy controls for federal information systems and organizations. Retrieved from http://nvlpubs.nist.gov/nistpubs/Special Publications/NIST.SP.800-53r4.pdf

Nimmer, C. (2015, Apr 17). How forge.mil changed the way the U.S. DoD developed software. Retrieved from https://opensource.com /government/15/4/how-forgemil-changed-way-DoD-develops-software

O'Connor, A., Ong, K. W., Sander, T., & Ferlo, M. (n.d.). Government policies on open source. Retrieved from https://courses.cs.washington.edu/ courses/csep590/04au/clearedprojects/Ferlo.pdf

O'Donohue, D. J. (2018). Cyberspace Operations Occupational Field (OCCFLC 17). Retrieved from http://www.marines.mil/DesktopModules/ ArticleCS/Print.aspx?PortalId=59&ModuleId=46529&Article=146900

Paul, M. (2014). The measurable and important benefits of DevOps. Retrieved from https://www.logicworks.com/blog/2014/10/measurable-important-benefits-devops

Perera, D. (2007, Oct 4). Is the Defense Travel System ready to fly? Retrieved from https://fcw.com/articles/2007/10/04/is-the-defense-travel-system-finally-ready-to-fly.aspx

Reed, D., Cohen, J., Majumder, A., Chonko, J., & Walker, J. (2013). The Coast Guard Machinery Control System (CGMCS): Commonality come true. Arlington, VA.: Thor Solutions, LLC.

Sabhlok, R. (2013, Jul 18). OSS: The hidden cost of free. Retrieved from https://www.forbes.com /sites/rajsabhlok/2013/07/18/open-source-software-the-hidden-cost-of-free/#521127734001

Saunders, M. (2014, May 9). How Munich switched 15,000 pcs from windows to Linux. Retrieved from https://www.linuxvoice.com/the-big-switch/

Scott, T., & Rung, A. E. (2016, Aug 8). *Memorandum for the heads of departments and agencies* [Memorandum]. Washington, DC: Department of Defense. Retrieved from https://obamawhitehouse.archives.gov/sites/default /files/omb/memoranda/2016/m_16_21.pdf

Serbu, J. (2018, Mar 15). Amid Congressional mandate to open source DoD's software code: Code.mil serves as guidepost. Retrieved from https://federalnewsradio.com/on-DoD/2018/03/amid-congressional-mandate-to-open-source-DoDs-software-code-code-mil-serves-as-guidepost/

Shachtman, N. (2010, Aug 31). HP Holds Navy Network Hostage for $3.3 Billion. Retrieved: from https://www.wired.com/2010/08/hp-holds-navy-network-hostage/

Snyk. (2017). The state of open source security. Retrieved from https://snyk.io/stateofossecurity/

United States Digital Service. (2016). *2016 Report to congress: Modernizing the Department of Defense travel system*. Washington, DC: Department of Defense. Retrieved from: https://www.usds.gov/report-to-congress/2016/defense-travel/

United States Navy Chief of Information. (2010). Navy stands up fleet cyber command: reestablishes U.S. 10th Fleet. Retrieved from http://www.navy.mil/ submit/display.asp?story_id=50954

Vaughan-Nichols, S. (2014a, Apr 7). Heartbleed: Serious OpenSSL zero-day vulnerability revealed. Retrieved from https://www.zdet.com/article/heartbleed - serious-openssl-zero-day-vulnerability-revealed/

Vaughan-Nichols, S. (2014b, Apr 14). Heartbleed: Open source's worst hour. Retrieved from https://www.zdet.com/article/heartbleed-open-sources-worst-hour/

Vaughan-Nichols, S. (2014c, Apr 16). Coverity finds OSS quality better than proprietary code. Retrieved from https://www.zdnet.com /article/coverity-finds-open-source-software-quality-better-than-proprietary-code/

Warren, T. (2018, Jan 4). Microsoft confirms it's acquiring GitHub for $7.5 billion. Retrieved from https://www.theverge.com/2018/6/4/17422788/ microsoft-github-acquisition-official-deal

WebSenor. (2018, Jun 22). Bringing Agility to Software Product Engineering with DevOps. Retrieved from https://www.websenor.com/bringing-agility-to-software-product-engineering-with-devops/.

Wennergren, D. (2009, Oct 16). *Clarifying Guidance Regarding OSS* [Memorandum]. Washington, D.C.: Department of Defense. Retrieved from https://DoDcio. defense.gov/Portals/0/documents/FOSS/2009OSS.pdf

Williams, M., & Rhodes, K. A. (2006, Sep) *Defense travel system: Reported savings questionable and implementation challenges remain* (GAO-06-980). Washington, DC: Government Accountability Office.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California