UCI-AM-17-041

ACQUISITION RESEARCH PROGRAM
SPONSORED REPORT SERIES

Achieving Better Buying Power for Mobile Open Architecture Software Systems through Diverse Acquisition Scenarios

1 May 2017

Dr. Walt Scacchi

Dr. Thomas A. Alspaugh

Institute for Software Research

University of California, Irvine

The research presented in this report was supported by the Acquisition Research Program of the Graduate School of Business & Public Policy at the Naval Postgraduate School.

Executive Summary

This research seeks to identify, track, and analyze software component costs and cost reduction opportunities within diverse acquisition life cycle scenarios for open architecture systems accommodating Web-based and mobile devices, where such systems combine best-of- breed software components and software products lines subject to different intellectual property license and cybersecurity requirements. Most large-scale government and business enterprises continually seek new ways to improve the functional capabilities of their software-intensive systems through lower acquisition costs. The acquisition of open architecture (OA) systems that can adapt and evolve through replacement of functionally similar Web-based and mobile device- based software components and software product lines (SPLs) is an innovation that can lead to lower cost systems, increased competition and innovation, with more powerful functional capabilities. OA system acquisition, development and deployment are thus seen as an approach to realizing Better Buying Power (BPP) goals for lowering system costs while jointly improving competition, adoption of OA systems that utilize standardized interfaces, utilize open source software (OSS) components where appropriate, increase small business roles and opportunities, use of technical development phase for true risk reduction and rapid prototyping, as well as doing more without more [Kendall 2015, Scacchi and Alspaugh 2015]. Finally, this acquisition research supports and advances a public purpose by investigating acquisition challenges arising from the adoption and deployment of secure OA software systems for Web-based or mobile devices, which is of contemporary concern to most academic, business, or government enterprises.

Our research objective was to develop new ways and means for identifying, tracking, and analyzing the costs and other BBP 3.0 opportunities associated with the acquisition life cycle of OA software systems. OA system software elements can include either OSS or proprietary CSS components subject to different IP licenses and cybersecurity constraints. Such components

may be configured into different, functionally similar versions that allow for common but costly CSS components to be replaced by their OSS counterparts, as a strategy to reduce software acquisition costs. Such replacement or substitution may arise at different stages of system acquisition including system design, integration, deployment, and evolution. *But it is unclear what happens within and across diverse acquisition scenarios that (a) seek to produce assembled capabilities for command, control, communications, cyber and business (C3CB) applications utilizing (b) OA software components are widgets, apps, or mashups that arise from (c) multi- party engineering efforts in heterogeneous software producer ecosystems, and where (d) Program offices or warfighters are expected to serve as system integrators.* Recent government and business enterprise policy encourages the move to component-based OA software systems, especially moves to embrace new mobile computing devices like tablets, smartphones, and Web- based software application services [Cochran and Reed14, Defense Information Systems Agency12a, Takai 2012] and better buying initiatives [Kendall 2015, Scacchi and Alspaugh 2014, Scacchi and Alspaugh 2015], informs us as well.

Web-based OA systems and mobile devices systems often integrate components independently developed by software producers using OSS or CSS, which then may be integrated into complete systems by system integrators [Cochran and Reed14, George, Morris, Galdoris, et al.14, Reed, Benito, Collens, Stein 2012, Reed, Benito, Collens, Stein 2012, Scacchi and Alspaugh 2014]**.** Acquisition personnel will increasingly be called on to review and approve security measures employed during the design, integration, deployment, and evolution of OA systems [Scacchi and Alspaugh 2013b, Scacchi and Alspaugh 2013c]. Our effort builds on related acquisition research efforts at the Software Engineering Institute (SEI) that address SPLs [BeJ10, JoB11], as well as on acquisition and development of secure OA systems built with widgets and apps [Conley, Brockman, DIreks, et al.14, Cochran and Reed14, George, Galdorisi, Morris, O"Neil14, George, Morris, Galdoris, et al.13, George, Morris, Galdoris, et al.14, Reed, Benito, Collens, Stein 2012, ReN14, Scacchi and Alspaugh 2014, Scacchi and Alspaugh 2014c]. Other related research addressing OSS [HiW10, Ke12, MarL11], component-based software ecosystems [Endres-Niggemeyer 13, Reed, Benito, Collens, Stein 2012, Reed, Benito, Collens, Stein 2012, Scacchi and Alspaugh 2012b, Scacchi and Alspaugh 2013c, Scacchi and Alspaugh 2014a, Scacchi and Alspaugh 2014b], and better buying initiatives [Scacchi and Alspaugh 2014, Scacchi and Alspaugh 2015] informs us as well.

Our research continues to demonstrate how complex OA systems can be acquired, designed, built, and deployed with alternative components and connectors resulting in functionally similar system versions, to satisfy overall system security requirements and individual system component intellectual property (IP) and cybersecurity requirements [Scacchi and Alspaugh 2013a, Scacchi and Alspaugh 2013b, Scacchi and Alspaugh 2013c], as well as surfacing new challenges for achieving better buying power that can decrease (or increase) software acquisition costs [Scacchi and Alspaugh 2014a, Scacchi and Alspaugh 2014b, Scacchi and Alspaugh 2015a]. The research results in this report help to identify, track, and analyze software acquisition and development practices associated with different types of Web-based and mobile software components including widgets, apps, and mashups. This may then help us to highlight opportunities to realize cost reduction and improve opportunities to realize better buying power. Our research results are applicable to most academic, business, or government enterprises that deploy complex information systems.

Next, our research results are documented in this Final Report and center on an analysis of six issues that we believe create new diverse kinds of acquisition scenarios for developing and deploying both conventional and mobile open architecture software systems, especially those that incorporate Web or mobile software applications as apps or widgets.

Last, our research results have been well received in presentations to different audiences, including academic and industry research groups, the larger Defense community, and the Federal Government more broadly. In particular, throughout 2016 our research results have been presented to audiences at the

2016 Acquisition Research Symposium (Monterey, CA). Other project activities that produced material results include our paper on *Recent Trends and Challenges Affecting the Acquisition of Open Architecture Software Systems*, prepared and disseminated to accompany our invited presentation at The Aerospace Corporation in Los Angeles, CA. This paper accompanied our invited presentation at The Aerospace Corporation on our research in this project titled *Emerging Research Issues in the Defense Open Architecture Ecosystem.* Similarly, our paper on *Issues in the Development and Implementation of Open Architecture Software Systems*, prepared and submitted for publication in CrossTalk: The Journal of Defense Software Engineering. This paper was accepted and will appear in 2017. Finally, another material result from this project was an invited half-day tutorial addressing *Beyond Open Architecture: Issues, Challenges, and Opportunities in Open Source Software Development*, which was presented at the 2016 International Conference on Global Software Engineering to an academic and industry audience held in Irvine, CA. As can been seen in these chapters, common and differentiated research results found in the chapters represent our efforts at reaching out to different audiences interested in our research, and what advice or guidance it may offer to such audiences.

Acknowledgement

This report was supported by grant #N00244-6-1-004 from the Acquisition Research Program at the Naval Postgraduate School, Monterey, CA. No endorsement, review, or approval implied. This paper reflects the views and opinions of the authors, and not necessarily the views or positions of any other persons, group, enterprise, or government agency.

THIS PAGE INTENTIONALLY LEFT BLANK

About the Authors

Walt Scacchi—is senior research scientist and research faculty member at the Institute for Software Research, University of California, Irvine. He received a PhD in information and computer science from UC Irvine in 1981. From 1981–1998, he was on the faculty at the University of Southern California. In 1999, he joined the Institute for Software Research at UC Irvine. He has published more than 200 research papers, and has directed 70 externally funded research projects. In 2011, he served as co-chair for the 33rd International Conference on Software Engineering—Practice Track, and in 2012, he served as general co-chair of the 8th IFIP International Conference on Open Source Systems. [wscacchi@ics.uci.edu]

Thomas Alspaugh—is a project scientist at the Institute for Software Research, University of California, Irvine. His research interests are in software engineering, requirements, and licensing. Before completing his PhD, he worked as a software developer, team lead, and manager in industry, and as a computer scientist at the Naval Research Laboratory on the Software Cost Reduction, or A-7, project. [thomas.alspaugh@acm.org]

THIS PAGE INTENTIONALLY LEFT BLANK

UCI-AM-17-041

Achieving Better Buying Power for Mobile Open Architecture Software Systems through Diverse Acquisition Scenarios

1 May 2017

Dr. Walt Scacchi

Dr. Thomas A. Alspaugh

Institute for Software Research

University of California, Irvine

Disclaimer: This material is based upon work supported by the Naval Postgraduate School Acquisition Research Program under Grant No. N00244-16-1-0004. The views expressed in written materials or publications, and/or made by speakers, moderators, and presenters, do not necessarily reflect the official policies of the Naval Postgraduate School nor does mention of trade names, commercial practices, or organizations imply endorsement by the U.S. Government.

THIS PAGE LEFT INTENTIONALLY BLANK

Table of Contents

Executive Summary 1

Chapter 1: Research Overview 3
  Introduction 5
  Relevance of Our Efforts to Acquisition Research and Practice 7
  Acknowledgements 9
  References 11

Chapter 2: Achieving Better Buying Power for Mobile Open Architecture Software Systems through Diverse Acquisition Scenarios 15
  Abstract 17
  Introduction 19
  Recent Scenarios for Acquisition of OA Software Capabilities 21
    OA, Application Program Interfaces (APIs), Closed Source Software (CSS) and Open Source Software (OSS) Components 23
  Application domain of interest: OA C3CB Systems with Web/Mobile Devices Utilizing Widgets/Apps 27
  OA Software Ecosystems within the Defense Community 29
      A. Tenets of MultiParty Engineering (MPE): 30
      B. Attributes of Adaptive Agile Ecosystems (AAE): 31
  Emerging Issues in Developing and Deploying OA C3CB Systems with Mobile Components within Different Acquisition Scenarios 41
      1. Unknown or Unclear OA System Solutions 41
      2. Heterogeneously Licensed OA Software Capabilities 44
      3. Cybersecurity for OA Software Capabilities 46
      4. Software Component Build, Release, Deployment (BRD) Processes 51
      5. Software Component Evolution Practices Transmitted Across the OA Ecosystem 53
      6. New Business Models for Acquisition of Software Components and Widgets 57
    Recommendations for Achieving Better Buying Power for Secure Mobile OA Software Systems with Widgets/Apps Across Diverse Acquisition Scenarios 58
    Addressing Better Buying Power Issues 59
    Improving and streamlining acquisition processes for secure OA systems 61
    Encourage the adoption of acquisition business models in open source formats 61

Encourage the development, (re)use, and refinement of open source models of acquisition processes....63

Develop and employ techniques for streamlining acquisition processes for secure OA systems....64

Discussion and Conclusions....69

References....75

Chapter 1: Research Overview

THIS PAGE INTENTIONALLY LEFT BLANK

Introduction

This research focuses on create a new approach to address Better Buying Power (BBP) challenges that can arise within diverse acquisition scenarios for software systems for the business and other enterprises, including the DoD and other government agencies [Kendall 2015]. Program managers, acquisition officers, and contract specialists will increasingly be called on to review and approve choices between functionally similar low or no cost open source software (OSS) components, and commercially priced closed source software (CSS) components, to be used in the design, implementation, deployment, and evolution of secure open architecture (OA) systems [Department of Defense and General Services Afministration13]. We seek to make this a simpler, more transparent, and more tractable process. Such a process must identify, track, and analyze software component costs throughout the system life cycle, and be easy to reuse for different system application domains, in order to realize cost reductions and improve acquisition workforce capabilities. Our recent research demonstrates how complex OA systems can be designed, built, and deployed with alternative components and connectors resulting in functionally similar system versions, to satisfy overall system security requirements and individual system component intellectual property (IP) and cybersecurity requirements [Scacchi and Alspaugh 2013a, Scacchi and Alspaugh 2013b, Scacchi and Alspaugh 2013c], as well as surfacing new challenges for achieving BBP that can decrease (or increase) software acquisition costs [Department of Defense 2015, Kendall 2015, Scacchi and Alspaugh 2014, Scacchi and Alspaugh 2015]. These results may then help us to highlight opportunities for cost reduction and BBP. Expected results will be applicable to government agencies and industrial firms, as well as to mission-critical command, control, communications, cyber and business (C3CB) system application capabilities.

Our research effort focused on performance of four concurrent research tasks. We briefly describe each research task then follow with an elaboration of our research description and the acquisition research questions we address.

Task 1: Investigate the interactions between software system acquisition guidelines and processes, and the cost consequences of alternative software system architectures incorporating different mixes of OSS and CSS widgets, apps, and mashup components subject to diverse acquisition scenarios employing shared acquisition agreements among multiple parties (e.g., different Program Offices) that seek to produce assembled capabilities for C3CB applications using secure OA components and SPLs [Scacchi and Alspaugh 2013a, Scacchi and Alspaugh 2013b, Scacchi and Alspaugh 2013c, Scacchi and Alspaugh 2015]. This entails exploring the balance between development, verification, and validation of software licenses and security rights in diverse acquisition scenarios, as well as the software widget, app, and mashup component/license costs while managing the development and evolution of OA systems at design-time, build-time, and release and run-time.

Task 2: Develop formal foundations for establishing acquisition guidelines program managers can use in diverse acquisition scenarios for reduced cost software-intensive systems that rely on development and deployment of secure OA systems using OSS widgets, apps, and mashups, as well as software product line (SPL) technology and processes [Scacchi and Alspaugh 2011, Scacchi and Alspaugh 2012, Scacchi and Alspaugh 2013a, Scacchi and Alspaugh 2013b, Scacchi and Alspaugh 2013c, Scacchi and Alspaugh 2014, Scacchi and Alspaugh 2015].

Task 3: Continuing to develop concepts contributing to the emerging design of an automated approach supporting acquisition of secure, component-based, and increasingly mobile OA systems by (a) determining their conformance to acquisition guidelines/policies, contracts, and related license management issues, and (b) giving future acquisition workforce support and insights to properly review, approve, and manage the acquisition of complex systems that incorporate cost-sensitive acquisition of secure OA systems and software widget and app components [Scacchi and Alspaugh 2011, Scacchi and Alspaugh 2012, Scacchi and Alspaugh 2013a, Scacchi and Alspaugh

2013b, Scacchi and Alspaugh 2013c, Scacchi and Alspaugh 2014, Scacchi and Alspaugh 2015].

***Task 4*:** Document the investigation, foundations, and results of the research in: (a) a Technical Report delivered within 30 days of project completion to the Technical Point of Contact at NPS; (b) a research paper presented at the *13th Annual Acquisition Research Conference*, in Monterey, CA, May 2016 [Scacchi and Alspaugh 2016]; (c) a progress report with the OSD sponsor via a video teleconference or other meetings at a time to be determined during the period of the award; and (d) related research venues and publications, including periodic research progress reports.

Results from the first three tasks are captured in the publications found in chapters 3-6, while the specific results for task 4 are found in chapter 2, and this overall Final Report. However, the (slide deck) presentation that accompanies our 2016 Acquisition Research Symposium paper found in chapter 2 is not included in this Final Report, as it is already available on the ResearchSymposium.org online portal, associated with this symposium.

Relevance of Our Efforts to Acquisition Research and Practice

Overall, through this research effort, we continue to seek to identify, track, and analyze ways and means for how to articulate, tailor, and streamline the process for diverse acquisition scenarios for secure OA systems that accommodate Web-based and mobile devices running widgets, apps, and mashups. We seek to do so in ways that focus on software cost drivers and highlight opportunities for cost reduction through alternative software components or system configurations. This investigation is therefore applicable to complex software elements used in many kinds of component-based OA software-intensive systems within business and academic enterprises, other non-governmental organizations, as well as DoD and other governmental organizations.

Finally, we note that academic institutions, government agencies, and most large-scale business enterprises continually seek new ways to improve the functional capabilities of their software- intensive systems through lower acquisition costs. The acquisition of OA systems that can adapt and evolve through replacement of functionally similar Web-based and mobile device-based software components and SPLs is an innovation that can lead to lower cost systems with more powerful, more agile functional capabilities. There is a significant need for sustained research that investigates the interplay and inter-relationships between (a) current/emerging guidelines for the acquisition of software-intensive systems, and (b) how secure, reusable software product lines [Mactal and Spriull 2012, Womble, Schmidt, Arendt, Fain 2011] that employ an OA incorporating OSS/CSS component products (e.g., widgets, apps, and mashups) and their production processes [Scacchi and Alspaugh 2013b], are essential to improving the buying power and cost-reduction effectiveness of software-intensive program acquisition efforts.

OA system acquisition, development and deployment are thus an approach to realizing better buying outcomes for lowering system costs while jointly enabling more competition through the adoption of OA systems that utilize standardized interfaces, utilize OSS components where appropriate, increase small business roles and opportunities, use of technical development phase for true risk reduction and rapid prototyping, as well as doing more without more [Scacchi and Alspaugh 2014a, Scacchi and Alspaugh 2015].

Last, we are grateful for the support and funding we have received that enabled our acquisition research to continue, and as documented in this Final Report. We welcome any comments or questions regarding any materials or concepts presented in this Report.

Acknowledgements

Preparation of this report and all work products therein resulted from a research grant, #N00244-16-1-0004 from the Acquisition Research Program at the Naval Postgraduate School, Monterey, CA. None of the content of this Final Report has been reviewed, approved, or endorsed by the ARP, NPS, US Navy, Department of Defense or any other government agency. The work presented is solely the responsibility of the authors.

THIS PAGE INTENTIONALLY LEFT BLANK

References

Alspaugh, T.A, Asuncion, H. and Scacchi, W. (2012). The Challenge of Heterogeneously Licensed Systems in Open Architecture Software Ecosystems, S. Jansen, S. Brinkkemper, and M. Cusumano (Eds.), *Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry,* Edward Elgar Publishing, 103-120, Northampton, MA.

Alspaugh, T.A, Scacchi, W., and Asuncion, H. (2010). Software Licenses in Context: The Challenge of Heterogeneously Licensed Systems, *J. Assoc. Info. Systems*, 11(11), 730- 755.

Cochran, J. and Reed, H. (2014). *DoD Widget Working Group Report*, 13 March 2014.

Conley, K., Brockman, B., Diercks, P., George, A., Lam, W., Lozano, A., Palnter, R. and Tolentino, G. (2014). Achieving Information Dominance: Unleashing the Ozone Widget Framework. *Proc. 19th Intern. Conf. Command and Control Research & Technology Symposium* (ICCRTS), Arlington, VA, paper 109. June.

Defense Information Systems Agency (2012). Strategic Plan: 2013-2018, Version 1.0, accessed 30 October 2012.

Department of Defense, *Better Buying Power*, http://bbp.dau.mil/ accessed 15 June.

Department of Defense and General Services Administration (2013). *Improving Cybersecurity and Resilience through Acquisition,* November 2013, accessed June 2015.

Department of Defense Open Systems Architecture (2011). *Contract Guidebook for Program Managers*, Vol. 0.1, December, https://acc.dau.mil/OSAGuidebook *Acquisition Research Symposium*. Vol. 1, 76-82, Naval Postgraduate School, Monterey, CA.

Endres-Niggemeyer, B. (2013). The Mashup Ecosystem, in *Semantic Mashups: Intelligence Reuse of Web Resources*, Springer, 1-50.

George, A., Galdorisi, G., Morris, M., and O'Neil, M. (2014). DoD Application Store: Enabling C2 Agility, *Proc. 19th Intern. Command and Control Research and Technology Symposium*, Paper-104, Alexandria, VA, June 2014.

George, A., Morris, M., Galdorisi, G., Raney, C., Bowers, A., and Yetman, C. (2013) Mission Composable C3 in DIL Information Environments using Widgets and App Stores. *Proc. 18th Intern. Command and Control Research and Technology Symposium*, Paper-036, Alexandria, VA, June 2013.

George, A., Morris, M. and O'Neil, M. (2014). Pushing a Big Rock Up a Steep Hill: Lessons Learned from DoD Applications Storefront, *Proc. 11th Annual Acquisition Research Symposium*, Vol. 1, 306-317, Naval Postgraduate School, Monterey, CA.

Guertin, N.H., Sweeney, R., and Schmidt, D.C. (2015). How the Navy Can Use Open Systems Architecture to Revolutionize Capability Acquisition: The Naval OSA Strategy Can Yield Multiple Benefits. *Proc 12th Annual Acquisition Research Symposium,* Monterey, CA, NPS-AM-15-004, May 2015.

Kendall, F. (2015). *Implementation Directive for Better Buying Power 3.0,* 9 April 2015.

Mactal, R., Spruill, N. (2012). A Framework for Reuse in the DoN. *Proc. 9th Acquisition Research Symposium,* Vol.1, 149-164, Naval Postgraduate School, Monterey, CA.

Reed, H., Benito, P., Collens, J., and Stein, F. (2012). Supporting Agile C2 with an Agile and Adaptive IT Ecosystem, *17th Intern. Command and Control Research and Technology Symposium* (ICCRTS), Paper-044, Fairfax, VA, June 2012.

Reed, H., Nankervis, J., Cochran, J., Parekh, R., and Stein, F. (2014). Agile, Adaptive IT Ecosystem: Results, Outlook, and Recommendations, *Proc. 19th Intern. Command and Control Research and Technology Symposium* (ICCRTS), Paper-011, Arlington, VA, June.

Scacchi, W. and Alspaugh, T., (2011). Advances in the Acquisition of Secure Systems Based on Open Architectures, *Proc. 8th Acquisition Research Symposium*, Vol. 1, Naval Postgraduate School, Monterey, CA.

Scacchi, W. and Alspaugh, T., (2012) Understanding the Role of Licenses and Evolution in Open Architecture Software Ecosystems, *J. Systems and Software*, 85(7), 1479-1494, July.

Scacchi, W. and Alspaugh, T., (2013a). Streamlining the Process of Acquiring Secure Open Architecture Software Systems, *Proc 10th Annual Acquisition Research Symposium,* Monterey, CA, 608-623, May 2013.

Scacchi, W. and Alspaugh, T. (2013b). Processes in Securing Open Architecture Software Systems, *Proc. 2013 Intern. Conf. Software and System Processes*, 126-135, San Francisco, CA, May 2013.

Scacchi, W. and Alspaugh, T. (2013c). Challenges in the Development and Evolution of Secure Open Architecture Command and Control Systems, *Proc. 18th Intern. Command and Control Research and Technology Symposium*, Paper-098, Alexandria, VA, June 2013.

Scacchi, W. and Alspaugh, T. (2014). Achieving Better Buying Power through Cost- Sensitive Acquisition of Open Architecture Software Systems. *Proc 11th Annual Acquisition Research Symposium,* Monterey, CA, NPS-AM-14-C11P07R01-036, May 2014.

Scacchi, W. and Alspaugh, T. (2015). Achieving Better Buying Power through Acquisition of Open Architecture Software Systems for Web and Mobile Devices. *Proc 12th Annual Acquisition Research Symposium,* Monterey, CA, NPS-AM-15-005, May 2015.

Scacchi, W. and Alspaugh, T. (2016). Achieving Better Buying Power for Mobile Open Architecture Software Systems Through Diverse Acquisition Scenarios, *Proc 13th Annual Acquisition Research Symposium*, Monterey, CA, SYM-AM-16-019, May 2016.

Takai, T.M. (2012). *Department of Defense Mobile Device Strategy*, Version 2.0, Office of the DoD Chief Information Officer, May 2012. http://www.defense.gov/news/dodmobilitystrategy.pdf accessed May 2013.

Womble, B., Schmidt, W., Arendt, M., and Fain, T. (2011). Delivering Savings with Open Architecture and Product Lines, *Proc. 8th Acquisition Research Symposium*, Vol. 1, 8- 13, Naval Postgraduate School, Monterey, CA.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2: Achieving Better Buying Power for Mobile Open Architecture Software Systems through Diverse Acquisition Scenarios

THIS PAGE INTENTIONALLY LEFT BLANK

Abstract

The U.S. Defense Community denotes an ecosystem of system or software component producers, system integrators and customer organizations. For a variety of reasons this community now embraces the need to utilize open source software (OSS) and proprietary closed source software (CSS) in the system capabilities or software components it acquires, design, develops, deploys, and sustains. But the long-term transition to agile and adaptive capabilities that integrate bespoke or legacy, OSS and CSS components, has surfaced a number of issues that require acquisition-research-led approaches and solutions. In this paper, we identify and describe six key issues now found in the Defense software ecosystem: (1) unknown or unclear software architectural representations; (2) how to best deal with diverse, heterogeneous software IP licenses; (3) how to address cybersecurity requirements; (4) challenges arising in software integration and release pipelines; (5) how OSS evolution patterns transform software IP and cybersecurity requirements; and (6) the emergence of new business models for software distribution and cost accounting. We use the domain of command and control systems under different acquisition scenarios as our focus to help illuminate these issues along the way. We close with suggestions for how to resolve them.

THIS PAGE INTENTIONALLY LEFT BLANK

Introduction

The U.S. Defense Community, which includes the military services and civilian-staffed agencies, is among world's largest acquirers of commodity and bespoke (custom) software systems. The Defense Community further extends its reach and influence on a global basis through national treaties and international alliances through enterprises like NATO. The Department of Defense (DoD), other government agencies, and most large-scale business enterprises continually seek new ways to improve the functional capabilities of their software-intensive systems while lowering acquisition costs. The acquisition of open architecture (OA) systems that can adapt and evolve through replacement of functionally similar software components is an innovation that can lead to lower cost systems with more powerful functional capabilities. OA system acquisition, development, and deployment are thus seen as an approach to realizing Better Buying Power (BPP) goals for lowering system costs, achieving technical excellence, enabling innovation, and advancing the acquisition workforce (Kendall 2015).

Bespoke software systems are produced and integrated within the Defense Community. In addition Defense system acquisition or procurement enterprises also obtain wares from most non-Defense industry providers of software systems, application or services (i.e., the mainstream software products or services industry). The acquisitions often entail software procurement or development contracts valued in the millions to hundreds-of-millions of dollars (Myers and Obendorf 2001). At this scale of endeavor and economic value, certain kinds of software engineering (SE) research problems arise that are not visible or are insignificant in smaller scale SE R&D efforts.

In this paper, we focus attention to the slice of this world that focuses on the development and deployment of software-intensive *command, control, communication, cyber and business* systems, (hereafter, C3CB). We further limit our focus to the most general software elements found in C3CB system capabilities, for example software infrastructure components, common development technologies supporting app/widget development, and mission-specific apps/widgets, in particular widgets produced with the Ozone Widget Framework (Conley, Brockman, Dierkes, et al. 2014). OWF (now called the *Ozone Platform* or OZP) was initially developed by the NSA, though is now identified as Government OSS (GOSS) and supported by a third-party contractor. OZP is widely used within the Defense and Intelligence Community. The growing importance of OZP within the Defense Community has directed focus to the production and integration of C3CB system capabilities to be assembled using it. This focus drives open discussion of and broad exposure to emerging research issues that arise from the production and integration (or software engineering—SE) of software components, and these in turn raise challenges for acquisition management and personnel. Specifically, we draw attention to issues surrounding the development, integration, and deployment of *multi-version and multi-variant software systems* composed from various open source software (OSS) and proprietary (CSS) software elements or remote services (Scacchi 2002, Scacchi 2010), eventually including recent efforts to support Web-compatible services and/or mobile devices in C3CB. This focus also provides exposure to future C3CB system capabilities composed from apps acquired through various acquisition regimes, including apps downloaded from different Defense Community App Stores (George, Morris, O'Neil, et al. 2013, George, Morris, Galdorisi, et al. 2014).

Recent Scenarios for Acquisition of OA Software Capabilities

Interest in open source software (OSS) within the U.S. Department of Defense (DoD) and military services first appeared more than 10 years ago (Bollinger 2003, Scacchi and Alspaugh 2008). More recently, it has become clear that the U.S. Defense Community has committed to a strategy of acquiring software-intensive systems across the board that require or utilize an “open architecture” (OA) which may incorporate OSS technology or OSS development processes that can help Defense customer organizations to achieve better buying power (Kendall 2015). Why? Among the reasons identified is the desire to realize more choices among software component producers or integrators, as producers and integrators often act in ways that lock-in their customer organizations to overly costly and sometimes underperforming and difficult to sustain systems.

One approach being explored focuses attention to agile and adaptive OA software components that are acquired and assembled (integrated) as C3CB system capabilities (*assembled capabilities or* AC) that are acquired and shared by multiple parties via independent “lines of efforts” acting within an ecosystems of producers, integrators and consumer organizations (Reed, Nankervis, Cochran et al. 2014; Scacchi and Alspaugh 2015). The goals of the AC approach include a shorter delivery and update cycle for mission components and an improved cybersecurity posture. We explain this approach as follows.

The AC approach contemplates independent acquisition lines of effort for different types of OA software components that can be acquired from independent providers:

- *Mission Component*s enable C3CB processes and present common operating picture data to end-users. Mission components may be realized as apps/widgets that may be deployed on mission-specific platforms, including those operating on secured Web/mobile devices.
- *Common Development Technology* provides AC development tools and common run-time applications servers that support the mission components. The servers are bundled with Shared Infrastructure, below.

- *Shared Infrastructure Components* combine local/remote application servers and data repositories with networking services and platforms.

Assembled capabilities therefore represent alternative configurations of mission-specific components that are produced with common development technology for deployment on shared infrastructure technology platforms.

Independent Lines of Effort (LOEs) by single or multi-party acquisition for mission components, common development technologies, or shared infrastructure components, are expected to greatly accelerate development and fielded deployment. This acceleration entails tradeoffs in increased dependency and risk management. Independent LOEs enable at least three alternative scenarios for acquiring OA C3CB system capabilities.

1. *Use current strategy and acquisition capabilities.* Here there is no focus on AC that utilize mission components, common development technologies, or shared infrastructure components.
2. *Augment deployed systems with mission components and common technologies.* Augmentation is either for (a) new mission functionality; (b) modernization “in place” so that part of the original system is deprecated as the new mission components are delivered; or (c) infrastructure replacement over parts of original system that may be combined with modernization efforts.
3. *Focus efforts on production, integration, security assurance, and deployment of mission components that use common technologies and shared infrastructure, and that can be assembled into different ACs.* This can entail production, integration, and delivery of all mission components in one contract vehicle; or alternatively the delivery of mission components partitioned across multiple acquisition contract vehicles, so as to spread and manage risk, while insuring multi-party buy-in commitment.

The following efforts provide examples where these alternative C3CB acquisition scenarios can be considered. First, the Air Force's Theater Battle Management Core System – Force Level (TBMCS-FL), which manages air tasking orders and airspace management among other things, is being harvested for current operational capabilities. These capabilities can then be encapsulated and delivered as mission components for other C3CB systems, using for example OZP widgets and supporting common technologies. For example, the C2AOS C2IS acquisition scenario intends to deliver harvested functionality as mission components. Air Force

AOC (Air Operations Center) is planning to include C2AOS C2IS as the replacement for TBMCS-FL, and will use the Navy ACS (hence indicating the need for multi-party acquisition agreements). This in turn implies the need for Joint C2, and needs to be copied to all Services. It represents an opportunity to reduce duplicate activities for producing equivalent C3CB system capabilities. Second, the Army's Distributed Common Ground System (DCGS-A) currently uses mission components for visualization (over 300 widgets available). DCGS-A will incorporate metadata mission components that utilize the DCGS Integration Backbone (DIB). Third and last, the Navy is deploying CANES and ACS (Agile Core Services) shared infrastructure to its fleet as a modernization effort (Guertin, Sweeney, Schmidt 2015).

There are now a number of policy directives within the Defense Community that formally recognize that OSS system elements can be treated as commercial-off-the-shelf (COTS) components, and that bespoke software system development projects will utilize an OA, unless otherwise justified and approved. Thus, developing contemporary C3CB that incorporate both OSS and new/legacy CSS elements denotes “business as usual.” However, many legacy Defense Community system capability producers are hesitant about how best to engineer such OA/OSS systems. For example, does an OA system imply/require that its software architecture be explicitly modeled, be accessible for sharing/reuse (e.g., as a Reference Model), and be modeled in a form/notation that is amenable to architectural analysis and computational processing (Wikipedia 2016)? Therefore, we can begin to identify what kinds of SE research issues can be observed and investigated within the Defense Community associated with its transition to OA systems and OSS software elements, specifically for Web and Mobile devices within the realm of C3CB.

OA, Application Program Interfaces (APIs), Closed Source Software (CSS) and Open Source Software (OSS) Components

OA C3CB system capabilities are assembled with mission components, common development technologies, and infrastructure. *Infrastructure components* are broadly construed to include non-mission specific software functionality or operations. Such components can include computer operating systems, web servers, database management systems, cloud services, mobile device management middleware, and others, along with desktop, mobile, or smartphone-based web browsers, word processors, email and calendaring, text/voice chat, and end-user media players. Example infrastructure components include the US Army's Common Operating Environment (COE), the Navy's Consolidated Afloat Networks and Enterprises Services (CANES) Afloat Core Services (ACS) (Guertin, Sweeney, Schmidt 2015), and similar elements in the Joint Intelligence Environment.

Common development technologies are common software development tools, libraries, or frameworks used to implement the necessary software functionality so that new or legacy mission components can be integrated into mission-specific software capabilities. Software technology frameworks (or common implementation libraries) like Oracle Java 8, Ozone Platform, OpenJDK (OSS Java Development Kernel for Android app development), and the NASA World Wind Java SDK; programming languages like Java or C++ and scripting languages like Javascript; may be utilized as common development technologies for developing mission components. Other software production capabilities like the Navy Tactical Cloud and CANES integrate both infrastructure and common development tools like Hadoop, MapReduce and other mission data analysis tools for the Tactical Cloud, and the Agile Core Services and Java for CANES.

Mission components represent a hybrid assortment of: (a) *simple widgets,* small, thin apps similar in spirit to those acquired and downloaded from online app stores (like a clock, calculator, dictionary, sticky note or unit converter); (b) *singular widgets,* more substantial functional components either created new (bespoke) or extracted from legacy systems that must run on a specific local computing platform (e.g., shipboard fire control system); or (c) *compound widgets,* hosted in a cloud and run as a remote cloud service over a single/multi-tiered client-server software architecture (e.g., Google Maps, NAlspaugh, Scacchi, Asuncion World Wind), and thus potentially accessible on a Web-based or mobile computing platform (e.g., mission-specific World Wind map viewer with hypermedia object overlays displayed on a Google Chrome web browser running on a secure Android mobile device).

OA seems to simply suggest software system architectures incorporating OSS/CSS infrastructure, common development technologies, and mission components that all utilize open application program interfaces (APIs). But not all software system architectures incorporating OSS/CSS components and open APIs will produce an OA, since whether an architecture is an OA depends on (a) how/why OSS/CSS and open APIs are located within it, (b) how OSS/CSS and open APIs are implemented, embedded, or interconnected within it, (c) whether the copyright (Intellectual Property) licenses assigned to different OSS/CSS components encumber all/part of the architecture into which they are integrated, and (d) choices among alternative architectural configurations and APIs that may or may not produce an OA (cf. Scacchi and Alspaugh 2008). This can lead to situations in which acquisition contracts stipulate a software-intensive system with an OA and OSS/CSS components, but the resulting software system may or may not embody an OA. This can occur when the architectural design of a system constrains the system requirements: if not all requirements can be satisfied by a given system architecture, if requirements stipulate specific types or instances of OSS/CSS (e.g., Web browsers, content management servers), if an architecture style (Bass, Clements, and Kazman 2003) is implied by given system requirements, or if requirements are implied by the choice to incorporate legacy software capabilities with one architectural style that are to be wrapped within mission-specific widgets with a different architectural style.

THIS PAGE INTENTIONALLY LEFT BLANK

Application domain of interest: OA C3CB Systems with Web/Mobile Devices Utilizing Widgets/Apps

C3CB are common information system applications that support modern military operations at a regional, national, or global level. These applications may be focused to address common military mission planning, mapping, resource status tracking and scheduling, mission performance, and monitoring activities through application sub-systems. However, closely related C3CB systems applications are also in common use within civilian/public safety agencies, public infrastructure/utility operations, live television and sports event broadcasting, massively multi-player online game operations centers, and even in international motorsports racing competition events like Formula 1. So the study of software production and system integration issues arising in the Defense Community can inform awareness of similar issues in other non-Defense software system domains, and vice versa.

Modern C3CB applications are increasingly expected/planned to be composed from best-available software components, whether OSS or CSS, utilizing bespoke or legacy software capabilities. Furthermore, as smartphones, tablets and laptop computers are being brought into the workplace, so too is interest increasing within the Defense Community in supporting the acquisition and development of Web-compatible widgets and mobile apps, provided through an emerging ecosystem of component producers and system integrators, for configuration into secure OA C3CB software system capabilities (George, Morris, Galdorisi, et al. 2014; Reed, Benito, Collens, et al. 2012; Reed, Nankervis, Cochran et al. 2014; Scacchi and Alspaugh 2013a; Scacchi and Alspaugh 2015). Common software elements for such systems include Web browsers open to extensions like custom mission-specific Map widgets, and remote content servers, email and calendaring, word processing, local/networked file servers, and operating systems. The data processed by the software may be of high-relevance to military missions/operations, or may just be the daily grind of data manipulated by “productivity” applications which most of us use routinely to perform/enact our work assignments. Security has been mostly addressed through system isolation or “air gaps” to the outside world due, for example, to airborne or afloat capability deployments. But this is no longer common practice, and cybersecurity concerns have risen to the top of functional and non-functional requirements for all such C3CB applications. New OA systems are now required to be secure by design, by implementation, and through release, deployment and evolution, as well as subject to independent testing and certification. Secure OA designs can then entail different schemes for encapsulating different (sets of) components, use of virtualization schemes, shims and wrappers, encrypting data transfers and storage, and configuring multi-level system access capabilities. But we have found examples in which different OA system designs and configurations propagate security obligations, and privacy protections and access rights are either mediated or nullified by different software component IP licenses or system updates.

OA Software Ecosystems within the Defense Community

In our view, a software ecosystem is a network of software component producers, system integrators, and customer organizations. In the Defense Community, producers and integrators are commonly industrial entities (defense contractors), while customer organizations are military program offices. Figure 1 presents an abstract view of a software ecosystem that associates software components or apps with their producers, system architectures with system integrators, and delivered component or integrated application systems with their customers. We also add annotations to indicate that each component or app has its own software IP license, and that integrated systems delivered to customers come with some composition of IP license obligations and rights propagated through the system's OA.

Multi-party acquisition and development system ecosystems – Many in the Defense community seek to embrace the acquisition and development of agile *command and control* (C3CB) and related enterprise systems [Agre, Gordon and Vasiliou 2014, George, Galdorisi, Morris, *et al*. 2014, George, Morris, Galdorisi, *et al*. 2013, Guertin and Womble 2012, Reed, Benito, Collens, *et al*. 2012, Scacchi and Alspaugh 2012b, 2013c, 2014a, 2016]. Such systems are envisioned to arise from the assembly and integration of system elements (application components, widgets, content servers, networking elements, etc.) within a software ecosystem of multiple producers, integrators, and customers who may supply or share the results of their efforts. The assembly and integration of system elements produces “assembled capabilities” (AC) for C3CB systems. Our purpose is to identify how our approach to the design of secure OA systems can be aligned with this emerging vision for agile C3CB system development and adaptive deployment. We also focus on design of OA system capability involving office productivity, Web and mobile device application components realized as widgets or apps [Agre, Gordon and Vasiliou 2014] that increasingly may be configured within secure AC [Scacchi and Alspaugh, 2011, 2012a, 2013b, 2015].

The design and development of agile C3CB systems follows from two sets of principles: one set addressing guidelines/tenets for multi-party engineering (MPE) of C3CB system components; the other set addressing attributes of agile and adaptive ecosystems (AAE) for producing AC or C3CB system elements [Reed, Benito, Collens *et al.* 2012, Reed, Nankervia, Cochran, *et al*. 2014, Scacchi and Alspaugh, 204a, 2014b, 2014c]. To help understand what we mean by a software ecosystem, we use **Figure 1** to represent where different parties are located across a generic software supply networks or multi-party relationships that emerge to enable the software producers to develop and release products that are assembled and integrated by system integrators for delivery to end-user organizations, via online storefronts [George, Galdorisi, Morris, *et al*. 2014, George, Morris, Galdorisi, *et al.* 2013].

For brevity, we identify the principles for MPE and AAE, as they are more fully explained elsewhere [Reed, Benito, Collens *et al.* 2012], but we do so in ways that foreshadow and more clearly align with our approach that follows in later sections.

A. Tenets of MultiParty Engineering (MPE):

- Provide small system components that can be rapidly developed, and accommodate different functionally equivalent variants, or functionally similar versions (software product lines).
- Certify components are consistent with “shared agreements” regarding security requirements, system architecture, data semantics, production and integration processes or process constraints, and other aspects of mission-specific or mission-common domain models.
- Supply diverse C3CB system components via a competitive marketplace of software component producers or system integrators.
- Assemble and integrate C3CB system capabilities from components available in the market that are consistent with relevant shared agreements.
- Provide feedback from C3CB system users to component producers or capability integrators to improve market efficiency and effectiveness.

B. Attributes of Adaptive Agile Ecosystems (AAE):

- Encourage and sustain a software ecosystem that is agile (supports assembly and integration C3CB capabilities) from components in market, and adaptive (supports substitution of functionally similar component versions or functionally equivalent component variants), in line with user feedback.
- Component markets are federated so as to accommodate sharing, reuse, or trading of components across system integrators or user organizations.
- Shared agreements serve as a basis for enabling multi-party collaboration in system development, integration, and evolution/sustainability.
- Production, integration, or post-deployment support for components or C3CB capabilities must be viable for small businesses or large, as well as promoting market diversity and effectiveness.
- customer/user organizations seek to manage portfolios of components or C3CB capabilities must be viable for small businesses or large, as well as promoting market diversity and effectiveness.
- customer/user organizations seek to manage portfolios of components or C3CB capabilities that collectively improve mission effectiveness, agility and adaptiveness, while reducing costs.

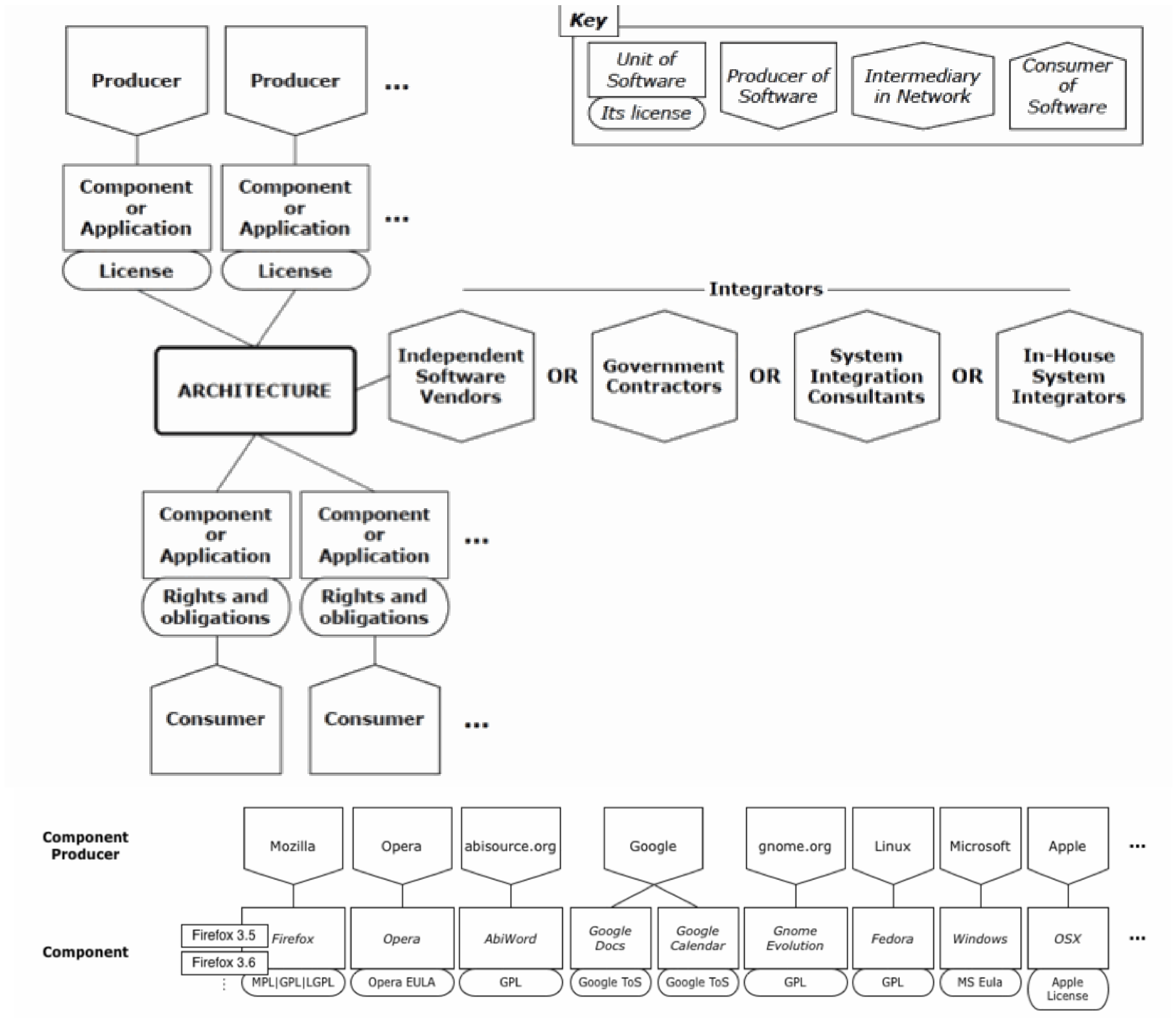


Figure 1. An abstract software ecosystem rendered as a network of software component producers, integrators of systems/AC, and end-user consumer organizations (upper part), along with a sample elaboration of producers, software component applications, and licenses for OA system components they employ (lower part).

As noted, OA system components can include software applications (apps) and widgets. Widgets are lightweight, single-purpose web-enabled applications that users can configure to their specific needs [Agre, Gordon and Vasiliou 2014, Gizzi 2011, George, Morris, Galdorisi, *et al.* 2013, Scacchi and Alspaugh 2013b, 2015]. Widgets can provide summary information or a limited view into a larger application that can be used alongside related widgets provides an integrated view, as required by users.

Moving towards shared development of Apps and Widgets as OA system components – Future agile C3CB OA systems may be configured by system integrators, end-user organizations, or war-fighters in the field. F**igure 2** outlines an increasingly common situation where two organizations (e.g., two Program Offices) have a shared interest in acquiring and deploying a specific mobile app for integration into an OA software system, and where multiple parties including software providers in industry may offer such a component as a browser-based widget, available in either CSS or OSS forms depending on the producer. App/widget acquisition would then be accomplished through access (perhaps free or paid per agreement among parties) to online, cloud-based repositories of software apps or user- interface widgets. **Figure 3** provides a recent sample of industry providers of different types of software components. The large size of this software producer ecosystem helps to underscore some of the challenges now facing organizations that want to achieve better buying power through sharing the acquisition or use of software apps or widgets.

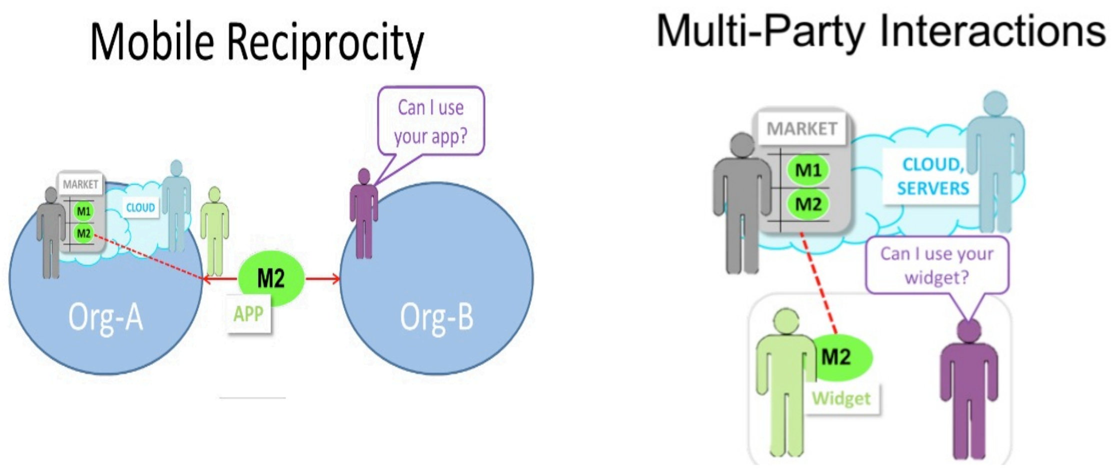


Figure 2: A recurring scenario where (left-side) two organizations want to reciprocally share use of a new mobile app/widget, that (right-side) must be acquired through interactions with multiple software component producers.

The Ozone Widget Framework (OWF) is government open source software (GOSS) effort that is central to such agile OA system development. The OZONE family of products includes the OWF and the OZONE Marketplace, the marketplace being an online repository whose operation is similar in kind to the online app stores by Apple and Google [Scacchi and Alspaugh 2013b]. These products are built to fit the needs of human centered fusion activities in network centric warfare environments. The OZONE family of products is designed as a presentation layer toolkit that can be rapidly deployed in a variety of mission contexts ranging from strategic planning to enable the creation of a real-time common operational picture and situation awareness applications. **Figure 4** displays examples of OWF-based widgets operating in a Web browser, while **Figure 5** shows OWF widgets deployed for use on a mobile device. **Figure 6** displays more commonly seen and experienced view of a desktop/laptop personal computer running a common Web browser with a word processing app/widget, while **Figure 7** shows the same word processing app/widget running on a common mobile device, a consumer smartphone.

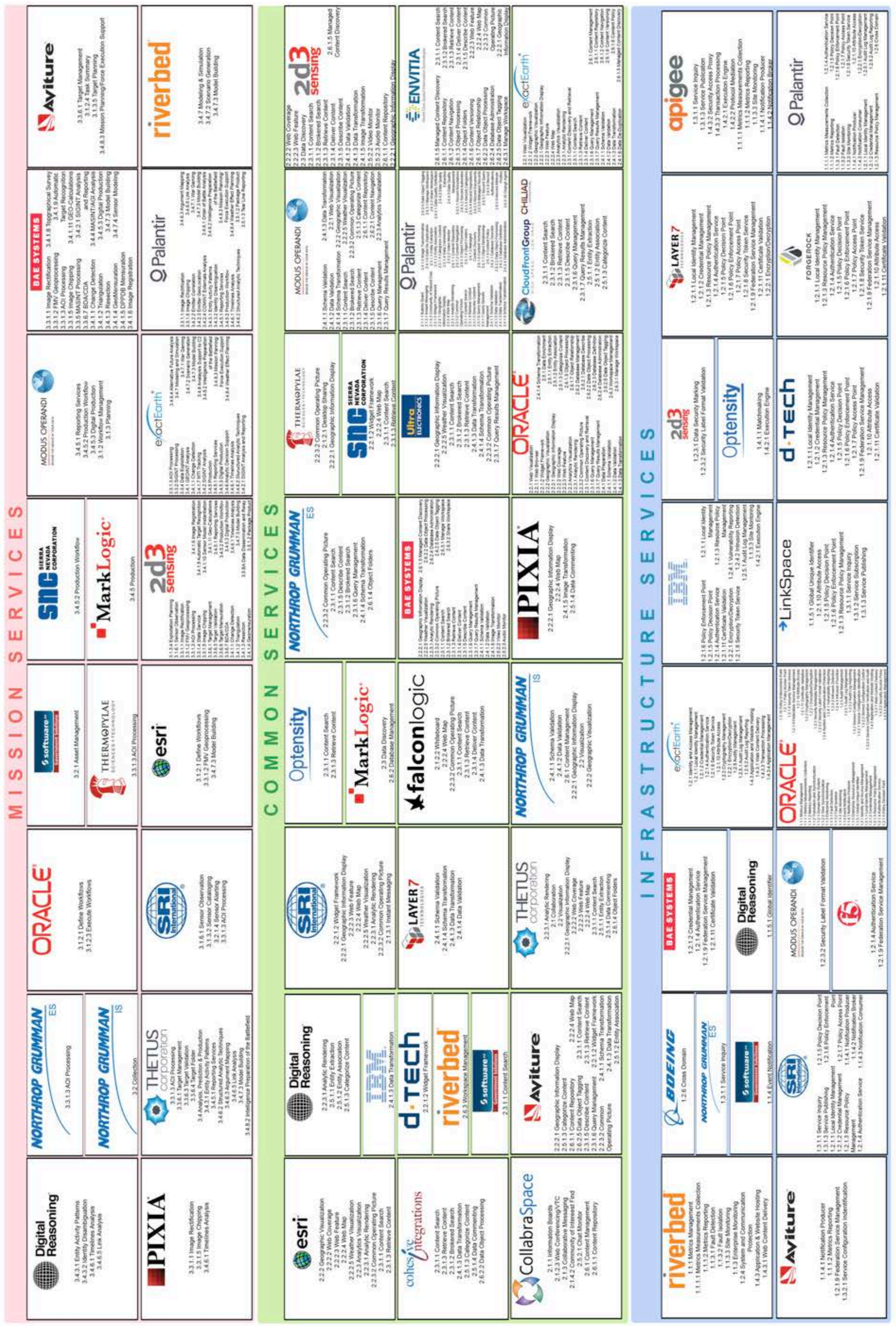


Figure 3. A sample of producers for mission components, common technologies, infrastructure components

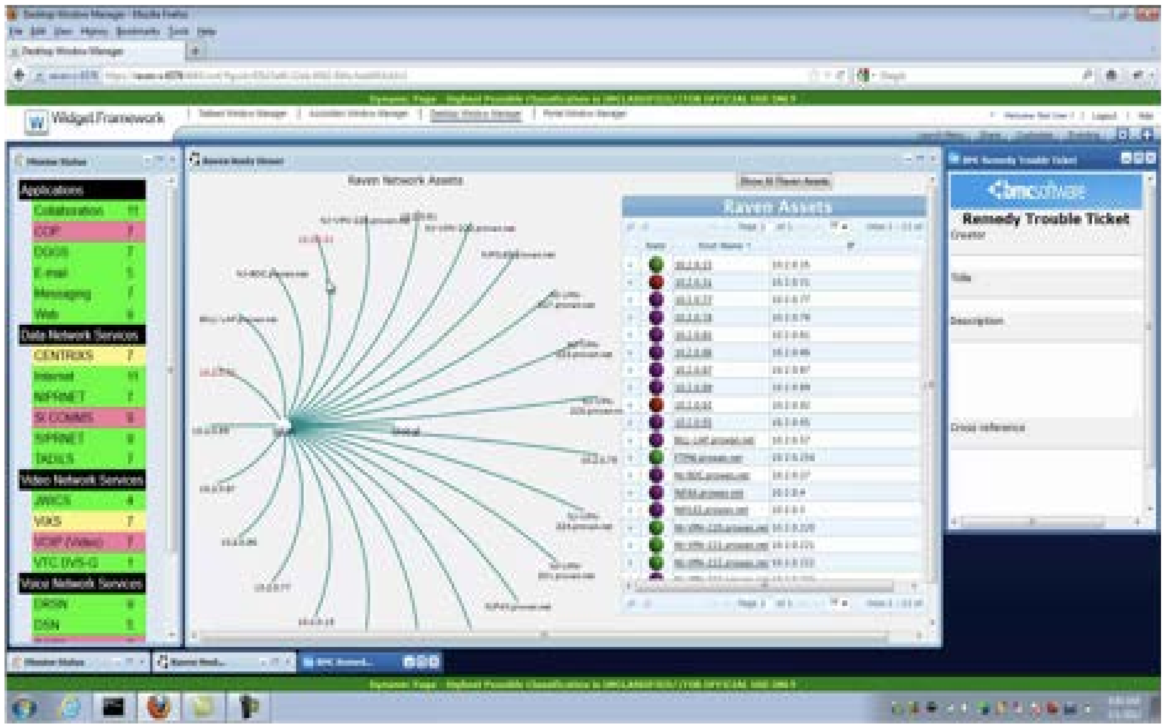


Figure 4. Ozone Widget Framework with simple, singular, and compound widgets running within a desktop/laptop personal computer Web browser.

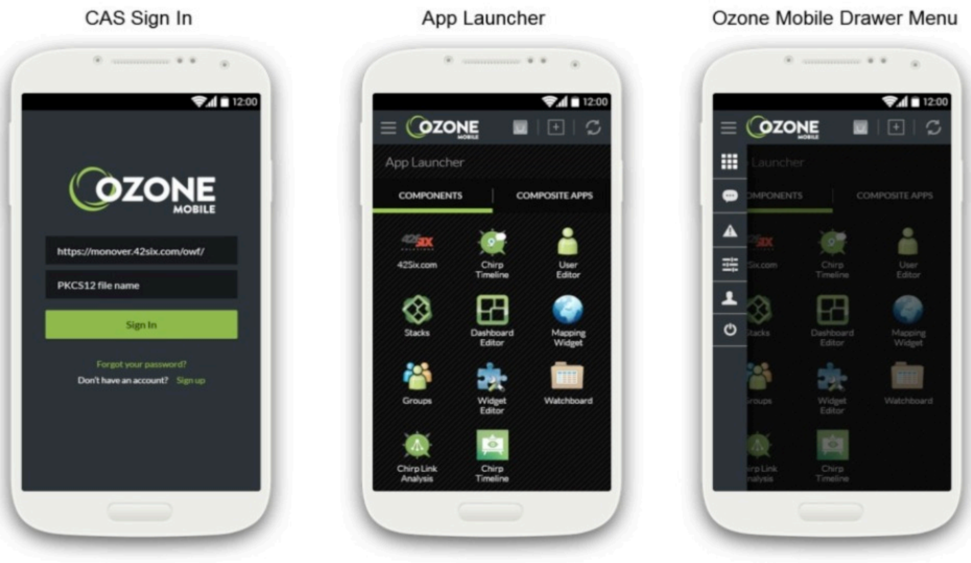
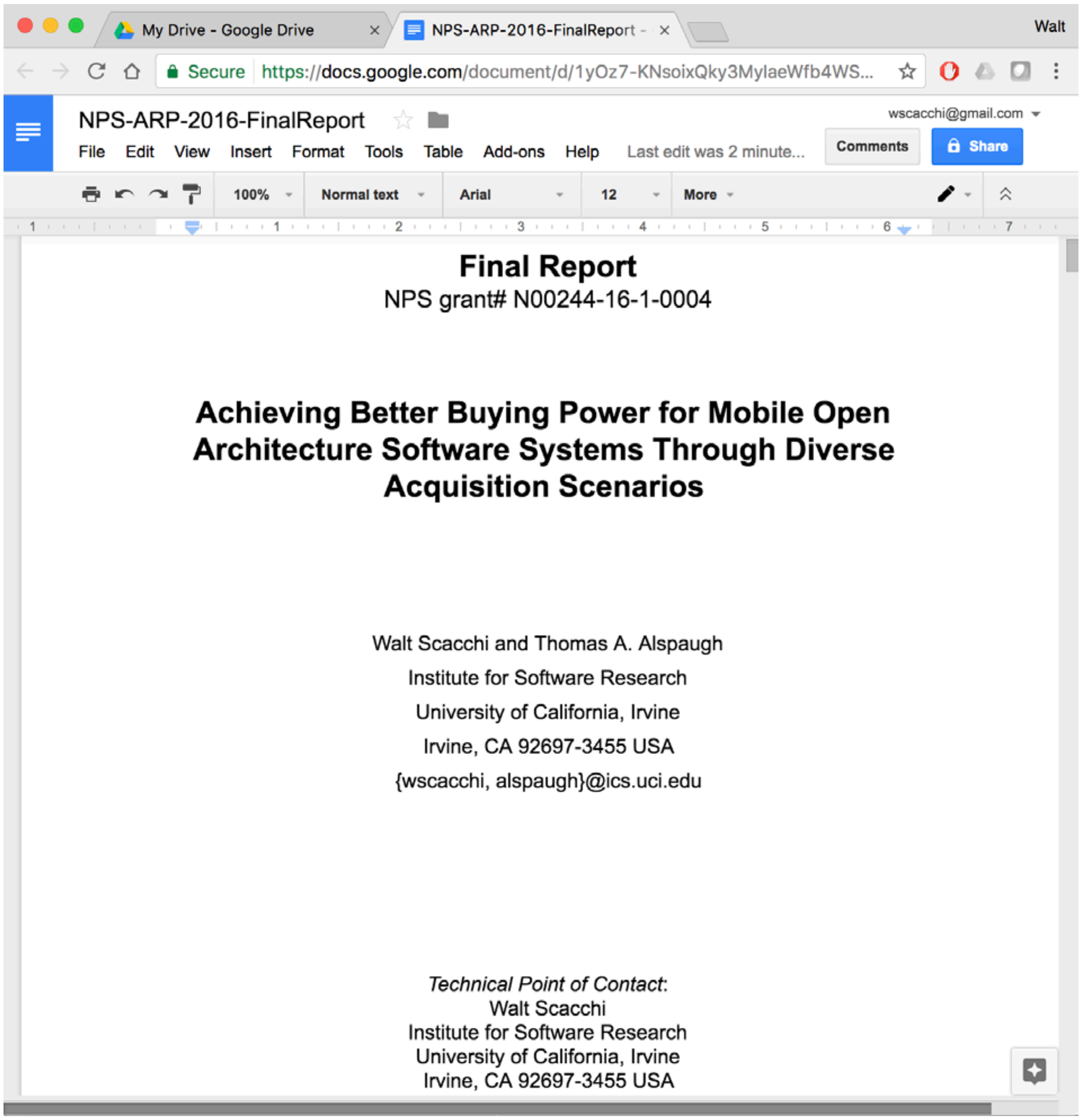


Figure 5. Ozone Widget Framework widgets of different kinds for use on a mobile device.



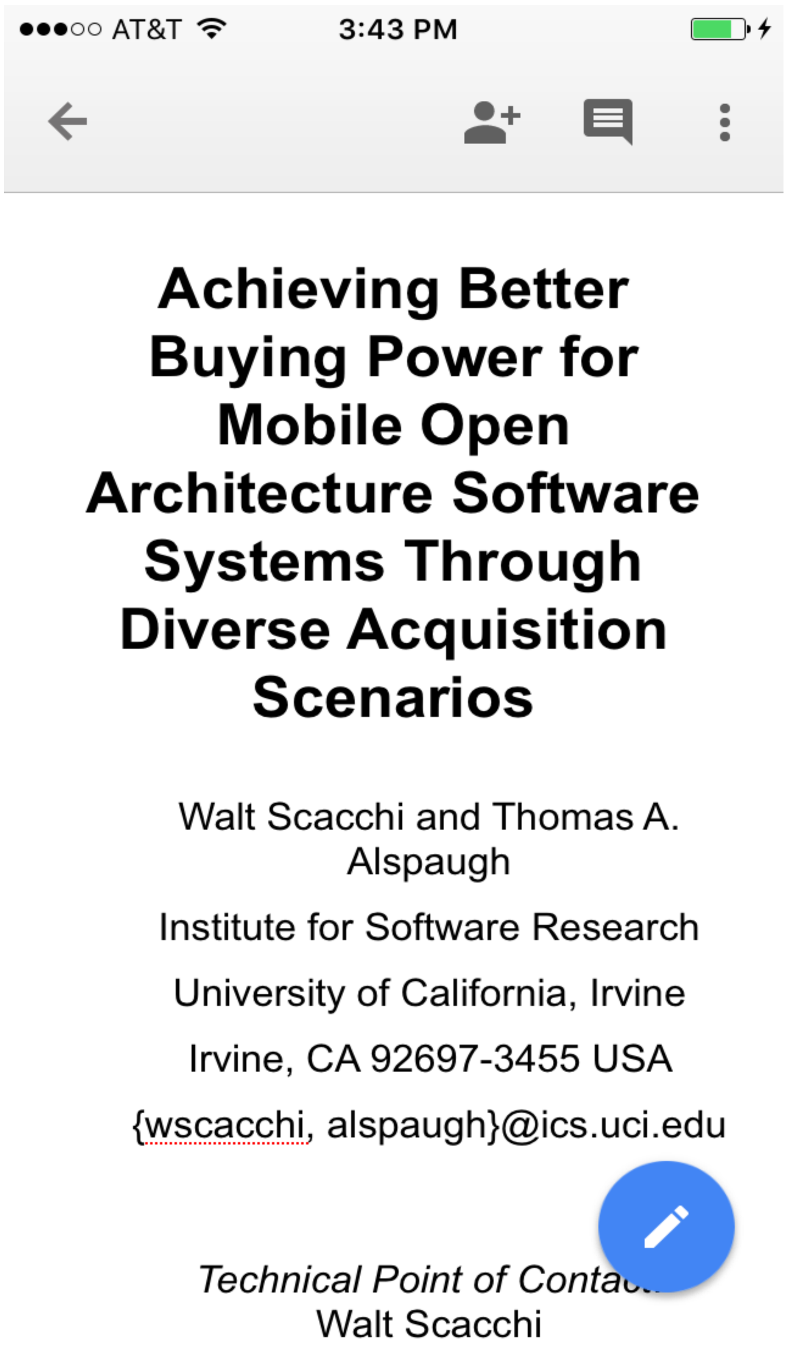
Final Report
NPS grant# N00244-16-1-0004

Achieving Better Buying Power for Mobile Open Architecture Software Systems Through Diverse Acquisition Scenarios

Walt Scacchi and Thomas A. Alspaugh
Institute for Software Research
University of California, Irvine
Irvine, CA 92697-3455 USA
{wscacchi, alspaugh}@ics.uci.edu

Technical Point of Contact:
Walt Scacchi
Institute for Software Research
University of California, Irvine
Irvine, CA 92697-3455 USA

Figure 6. A common personal computer Web browser (*Google Chrome*) shown running a singular word processing widget (*Google Docs*) editing a remote, cloud-based document (accessed via *Google Drive*) across a secure network data transfer connection (*Secure https*).



Achieving Better Buying Power for Mobile Open Architecture Software Systems Through Diverse Acquisition Scenarios

Walt Scacchi and Thomas A. Alspaugh

Institute for Software Research

University of California, Irvine

Irvine, CA 92697-3455 USA

{wscacchi, alspaugh}@ics.uci.edu

Technical Point of Conta

Walt Scacchi

Figure 7. Mobile device (smartphone) user display shown running singular widget for word processing (via *Google Docs Mobile*) the same document seen in Figure 6. Note the layout of the document content is reformatted by this widget to accommodate the mobile device's display properties.

To reiterate, there is growing interest within the Defense Community in transitioning to acquiring complex software system capabilities via an agile and adaptive ecosystem (Reed, Benito, Collens, et al. 2012; Reed, Nankervis, Cochran et al. 2014; Scacchi and Alspaugh 2015), where components may be sourced from alternative producers or integrators, allowing for more competition, and ideally lowering costs and improving the quality of software elements that arise from a competitive marketplace (Kendall 2015). But this adaptive agility to mix, match, reuse, mashup, swap, or reconfigure integrated systems, or to accommodate end-user architecting (Garlan, Dwivedi, Ruchkin, et al. 2012) as in-house integrations of mission components, requires that systems be compatible with or designed to utilize an OA.

Consequently, we can identify six kinds of emerging research challenges or issues for software capability acquisition that we have observed within the U.S. Defense Community as they move to produce, integrate, deploy and evolve OA systems for C3CB system capabilities that utilize contemporary OSS and bespoke/legacy CSS components. These issues center around (1) unclear representations of OA software system capabilities; (2) how best to accommodate diverse intellectual property licenses when combining bespoke/legacy OSS/CSS mission components; (3) how to accommodate diverse and complicated cybersecurity requirements; (4) technical challenges arising from alternative ways to integrate and deploy diverse software components; (5) how to accommodate many different paths within the Defense Community that drive software component evolution; and (6) how to estimate and manage the costs of acquiring, deploying, and sustaining diverse software-based mission components and C3CB system capabilities. These are examined in the next section.

With this background and sets of concepts for understanding a simplified view of the world of C3CB software systems, we now turn to identify and examine a set of issues that are now recurring in the acquisition, design, development, and deployment of such systems.

THIS PAGE INTENTIONALLY LEFT BLANK

Emerging Issues in Developing and Deploying OA C3CB Systems with Mobile Components within Different Acquisition Scenarios

There are at least six kinds of emerging acquisition research issues for software capability acquisition that we have observed within the U.S. Defense Community as it moves to OA systems for C3CB system capabilities [Scacchi and Alspaugh 2016]. These issues arise during the development, deployment and evolution phases of OA system acquisition.

1. Unknown or Unclear OA System Solutions

An OA entails a documented representation of software capability described in an architectural description language that specifies component types, component interconnections and connector types, open APIs, and their properties and interrelationships. The common core of a C3CB system OA resembles most enterprise business systems, as C3CB are a kind of management information system for navigating, mapping, tracking resources; scheduling people and other resources; producing plans and documentation; and supporting online email, voice or video communications. **Figure 8** depicts an OA representation for such a kind of system. This OA representation can be read as a “reference model” for a C3CB software product line (Womble, Schmidt, Arendt, et al. 2011).

Figure 9 further expands the sub-architecture of software components that denote configurations of mission-specific components as widgets. Thus, C3CB system capabilities can compose or reuse multiple or nested OA reference models.

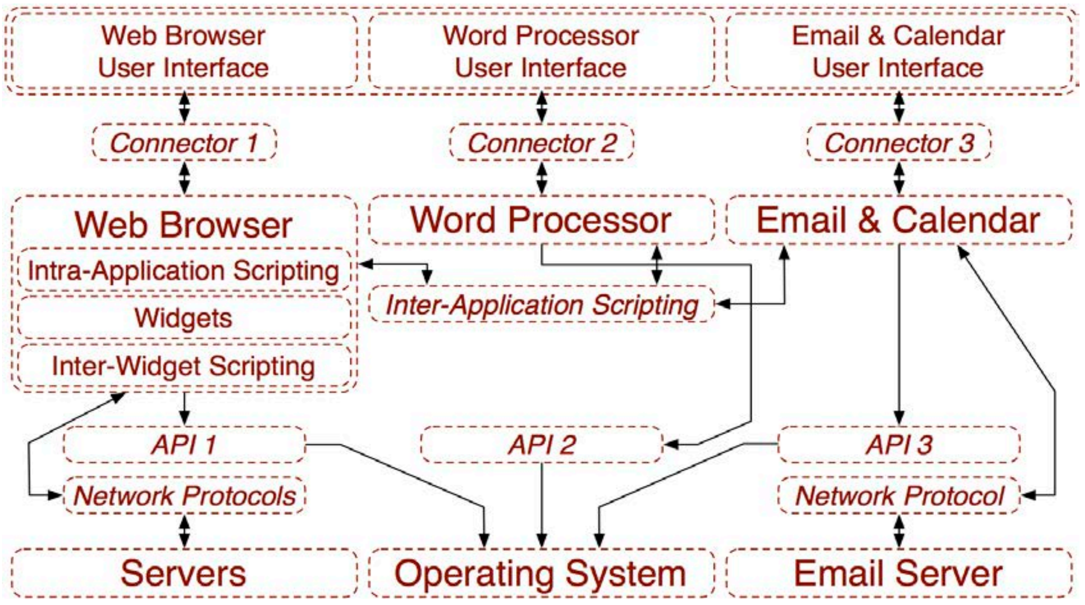


Figure 8. An OA reference model for common software component types including widgets interconnected within integrated C3CB system capability. Components come from producers that are assembled into OA C3CB capabilities by system integrators.

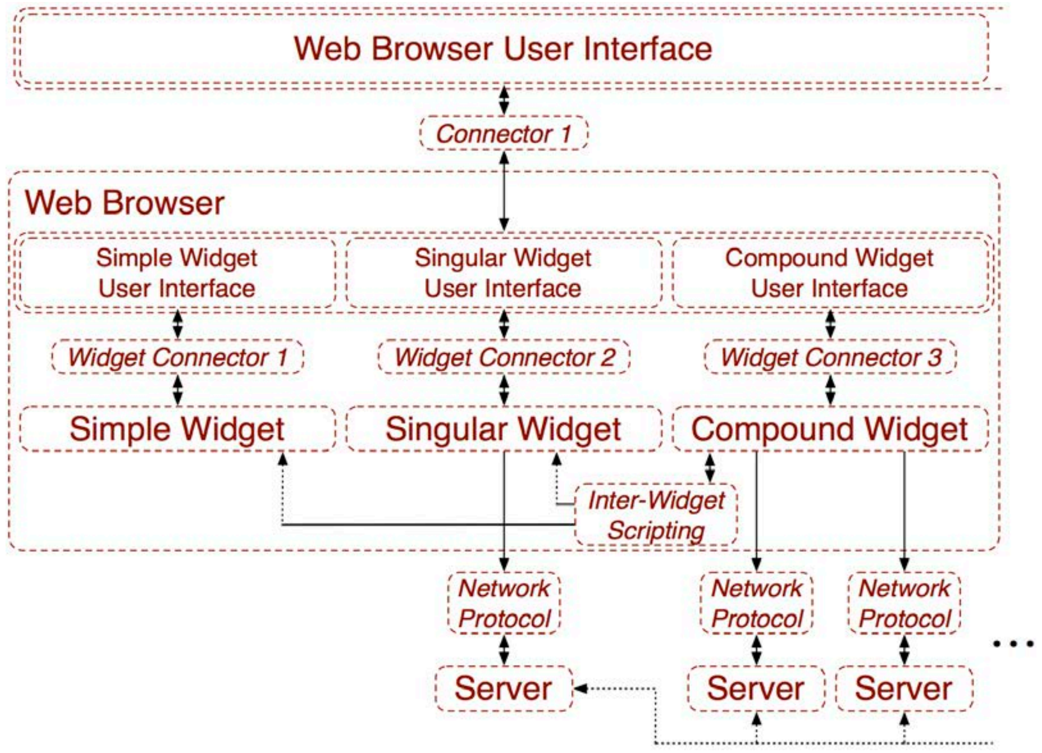


Figure 9. An OA reference model for common types of software widget components that can be connected and integrated to realize mission-specific C3CB system capabilities, within the overall OA shown on the left-side in Figure 8. Servers may be secured Web content servers, app servers, databases, or file system servers/repositories either available as legacy systems, or from software producers like those identified in Figure 3.

The next piece of the OA challenge we are studying is the envisioned transition with the Defense Community to C3CB system capabilities being composed by end-user system integration architects (Garlan, Dwivedi Ruchkin, et al. 2012) working within/for customer organizations, or potentially extended by end-users deployed in the field. This is the concept that surrounds the transition to discovering software components, apps, or widgets in Defense customer organization *app stores* (George, Morris, O'Neil, et al. 2013; George, Morris, Galdorisi, et al. 2014). These app stores are modeled after those used in distributing and acquiring software apps for Web-based or mobile devices, operated by Apple, Google, Microsoft, and others. How the availability of such Defense mission capability app stores will transform the way C3CB systems are produced, or whether they will be produced by legacy Defense industry contractors, remains to be seen. Said differently, how app stores transform OA software ecosystem networks, business models, and cybersecurity practices is an emerging challenge for acquisition and SE research in the Defense Community.

Another kind of challenge arises when acquiring new or retrofitting legacy C3CB software system applications that lack an open or explicit architectural representation identifying major components, interfaces, interconnections and remote services (if any). Though OA reference models and architectural description languages are in use within the SE research community, contemporary C3CB generally lack such descriptions or representations that are open, sharable, or reusable. This may be the result of legacy business practices in the Defense Community that see *detailed software architecture representations as proprietary IP* rather than as open, sharable technical data, even when OSS components are included or when applications sub-systems are entirely made of OSS code. An alternative explanation reveals that complex software systems like common Web browsers (Mozilla Firefox, Google Chrome, Apple Safari, Microsoft Internet Explorer) have complex architectures that integrate millions of SLOC that are not well understood, and that entail dozens of independently-developed software elements with complex APIs and IP licenses that shift across versions (Scacchi and Alspaugh 2012). For such systems the effort to produce an explicit OA reference model is itself a daunting architectural discovery, component/sub-system extraction, restructuring/refactoring, and continuous software evolution task (Choi and Scacchi 1990; Kazman and Carriere 1998). Thus, new ways and means for extracting software components interconnections and interfaces and transforming them into higher-level architectural representations of mission-specific apps/widget configurations are needed.

Harvesting legacy source/executable binary code entails many software engineering challenges that constrain acquisition efforts. First, legacy code provides too much technical detail and comparatively little abstraction of overall system configuration, composition, components and interconnection/dependencies. Second, incongruent computational system models (e.g. legacy data-flow versus publish-subscribe widgets) or hybrid OA AC arise when transitioning legacy system software elements into new widget-based mission components. Third, there is a general inability to visualize or analyze (test, selectively execute, translate into another programming language, etc.) overall system configurations, interconnections, or interfaces. Fourth, lacking these three, the potential for general software reuse is limited to executable code reuse, which is the lowest common denominator for reuse. Such reuse results in substantial blocks of unused code that cannot be easily removed due to indiscernible interdependencies. Last, when configuring mission components that entail legacy C3CB software applications wrapped for integration as widgets, different architectural styles can inadvertently be mixed (e.g., dataflow architecture for legacy C3CB software, and publish-subscribe architecture for configured mission widgets), which in turn raises the potential for architectural mismatches (Velasco-Elizondo, Dwivedi, Garlan et al. 2013) that may be difficult to determine or detect during system integration, especially when such integration activities are performed by end-user/consumer organizations.

2. Heterogeneously Licensed OA Software Capabilities

OSS components are subject to widely varying copyright, end-user license agreements, digital civil rights, or other IP protections. The Open Source Institute recognizes dozens of OSS licenses are in use, though the top 10 represents more than 90% of the open source ecosystem (Scacchi and Alspaugh 2012). This is especially true for OSS components or application systems that incorporate source code from multiple, independent OSS development projects, such as found in contemporary Web browsers like Firefox and Chrome which incorporate components from dozens of OSS projects, most with diverse licenses (Scacchi and Alspaugh 2012). This means that C3CB system capabilities that entail configuration of OSS/CSS components are subject to complex software IP obligations and rights that may defy tracking, or entail contradictory legal obligations or rights (Alspaugh, Scacchi and Asuncion 2010). Determining overall IP obligations for such systems is generally beyond the scope of expertise for software developers, as well as most corporate lawyers. Furthermore, we have observed many ways in which IP licenses interact within an OA software system, such that different architectural design choices that configure the same set of software components result in different overall system obligations and rights. Understanding multiple license interaction and IP mismatches is far too confusing for most acquisition professionals and Program Office decision-makers and a source of legal expense, or alternatively expensive indemnification insurance policies by the software producers or system integrators.

One complication that can be anticipated here arises when component types are replaced with versioned component instance alternatives (Scacchi and Alspaugh 2012). Consider the situation where a Web Browser (e.g., *Firefox 40.0.3* or *Chrome 47.0.2526.111 (64-bit)*; etc.) component has a specific IP license (e.g., *Mozilla Public License 2.0* or *GPL 3.0*) associated with the versioned instance, which in turn may be viewed by system integrators as enabling/limiting an integrated system's architectural design, depending on how different components are interconnected in ways that may or may not propagate (un) desirable IP obligations and rights—a concern that arises frequently when using components subject to the GPL (Scacchi and Alspaugh 2008). As we have learned in practice, corporate lawyers employed by Defense contractors or in government agencies do not have solutions for how to resolve such complexities, except via costly overall liability indemnification schemes, and efforts to distribute integrated systems with many IP obligations and few rights that effectively make an integrated open source system closed. This in turn can defeat the potential opportunities and benefits for commitment to OA systems that integrate OSS components.

Bespoke/legacy software components for OA AC design, integration and delivery within widgets will be subject to their bespoke/legacy IP obligations. This may include limits on the right to extract, restructure, or reengineer their architecture (cf. Choi and Scacchi 1990; Kazman and Carriere 1998) into open source formats. Similarly, IP licenses associated with OSS or new CSS components may impinge on their integration with these legacy components, or may limit disclosure of their interfaces that would allow more open integration of alternative software AC configurations developed by different Defense Community component producers [Scacchi and Alspaugh 2012).

Nonetheless, in our view, OA software ecosystems are defined, delimited, and populated with *niches* that locate specific integrated system solutions (Scacchi and Alspaugh 2012). Furthermore, we see that these niches effectively have *virtual IP licenses* that must be calculated via the obligations and rights that propagated across integrated system component licenses via union, intersection, and subsumption relations among them (Alspaugh and Scacchi 2012). Such calculation may appear to be daunting, and thus begs for a simpler, tractable, and computationally enforced scheme that can scale to large systems composed from many components, as well as be practically usable by C3CB system capability producers, integrators and acquisition professionals. In such a scheme, OSS/CSS licenses could formalize IP obligations as operational requirements (i.e., computationally enforceable, at the integrated system level) instantiated by system integration architects (Alspaugh, Scacchi and Asuncion 2010; Alspaugh and Scacchi 2013). Similarly, customer/user rights are then non-functional requirements that can be realized and validated as access/update capabilities propagated across the integrated system (Alspaugh and Scacchi 2013).

3. Cybersecurity for OA Software Capabilities

New types of software components like apps and widgets must be developed, deployed, and sustained in ways compatible with existing cybersecurity requirements. They must also be later adapted to accommodate emerging cybersecurity requirements that are not yet apparent. For example, there is growing interest in accommodating not just mobility, but also “Bring Your Own Device” (BYOD) capabilities.

BYOD suggests that end-users and war fighters are bringing their own mobile devices with themselves into the field to support their mission. However, BYOD clearly exacerbates the technical challenges of cybersecurity assurance, often in ways that cannot be readily anticipated, as when independently developed components co-evolve in conflict to one another [Weir 2014]. Nonetheless, acquisition policy demands that cybersecurity vulnerability and exposures be addressed [Defense Acquisition Guidebook 2015]. But at present, it is unclear what new kinds of requirements these new OA system components bring to the acquisition workforce. For example, a move to adopt mobile apps and/or mobile widgets means these OA system components must pass through an application security process for “vetting” these components.

Vetting entails establishing what cybersecurity requirements are to be verified, how they are to be validated, and where, when and by whom these activities should be performed. One approach is to assume a centralized authority can perform the vetting, such as by the operator of the Ozone Marketplace. But it is not clear there will ever only be one such authority. Instead, if we foresee multiple marketplaces, which are already appearing both in GOSS and industrial online settings, then the acquisition workforce will be challenged in how best to determine which cybersecurity requirements must be addressed, validated, and compliance certified, as well as by whom and how often. Consider the example, seen in **Figure 10**, of a widget for “emergency response incident command system,” developed for the Department of Homeland Security [Rockwell 2015]. How do its components (possibly GOSS) compare or interoperate with widgets or C3CB capabilities from DoD agencies or program offices concerned with C3CB system interoperability or C23CB assembled capabilities?

A move to widgets also presents new kinds of cybersecurity challenges when two or more widgets are configured together with one or more apps to create a mashup, providing an agile system capability. This situation refers to the technical challenges of inter-widget communication. Such component-component communication can be technically realized in different ways, such as via ad hoc, “open standards,” or publish-subscribe messaging interfaces, as well as point-to-point or as configured through a dynamic processing mashup [Chudnovsky, Fischer, Gaedke, *et al.* 2013, Endres 2013]. While OA system may rely on “open standards” style widget interfaces and communications patterns, widget communication/interface standards/interfaces are still very new technologies and techniques. Thus, it is unclear which will survive and be widely adopted [Endres 2013a].

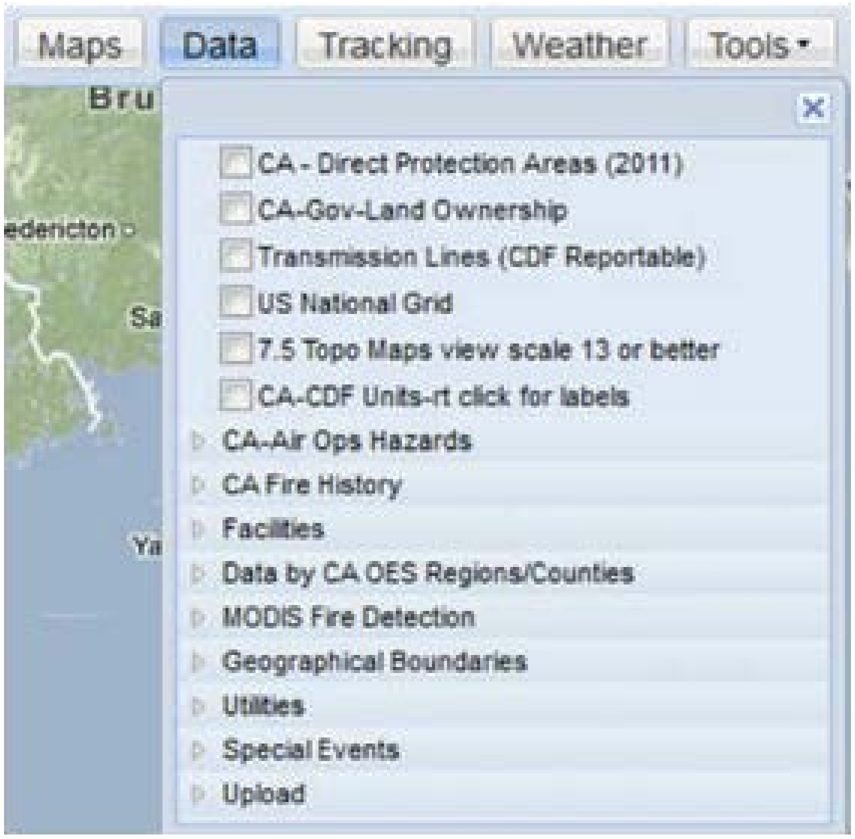


Figure 10. Example of a simple C3CB widget from the Next-Generation Incident Response System for Department of Homeland Security [Rockwell 2015].

Cybersecurity is a high priority requirement in all C3CB systems, applications, AC and platforms (Scacchi and Alspaugh 2013c; Scacchi and Alspaugh 2013d). No longer is cybersecurity something to be addressed after C3CB systems are developed and deployed—cybersecurity must be included throughout the design, development, deployment, and evolution of C3CB. However, the best ways and means for addressing cybersecurity requirements are unclear, and oftentimes somewhat at odds with one another depending on whether cybersecurity capability designs are specific to a: C3CB platform (e.g., operating system or processor virtualization; utilization of low-level operating system access control or capability mechanisms); component producer (secure programming practices and verification testing); system integrator (e.g., via use secure data communications protocols and data encryption); customer deployment setting (mobile: air-borne or afloat; fixed: offices, briefing rooms, operations centers); end-user authentication mechanisms; or acquisition policy (e.g., reliance on third-party audit, certification, assurance of system cybersecurity). However, in reviewing these different arenas for cybersecurity, we have found that the cybersecurity requirements or capabilities can be expressed in much the same way as IP licenses: using concise, testable formal expressions of obligations and rights. Some examples follow (capital letters are placeholders that denote specified system, service, or component contexts).

- The obligation that a user must verify his/her authority by password or other specified authentication process.
- The obligation for all components connected to specified component C must grant it the capability to read and update data in compartment T.
- The obligation to reconfigure a system in response to detected threats or known vulnerabilities, when given the right to select and include different component versions, or executable component variants.
- The right that a user or software component may read and update data in compartment T using the licensed component.
- The right that may allow replacement of a specified component C with some other vetted component.

These examples show how cybersecurity requirements can be expressed or paraphrased in restricted natural language (e.g., using a domain-specific language) into composite specifications that denote "security licenses" (Alspaugh, Scacchi and

Asuncion 2010; Alspaugh and Scacchi 2012). In this way, it should be possible to develop new software analysis tools whose purpose is to interpret cybersecurity obligations as operational constraints (executable) or provided capabilities (access control or update privileges), through mechanisms analogous to those used for analyzing software licenses (Alspaugh, Scacchi and Asuncion 2010; Alspaugh and Scacchi 2012), and how component or sub-system-specific obligations and rights can be propagated across a system's architecture.

We similarly envision the ability for OA system capabilities to be produced and integrated according to different cybersecurity requirements, depending on where and how they are deployed [Scacchi and Alspaugh 2013d). For example, in Figure 11 we show one possible layout of software components that confines different sub-configurations within different virtual machines, where these virtual machines may also be hierarchically nested (as is the case when mission-specific widgets that entail legacy C3CB applications must be securely confined during run-time to access remote servers, that are distinct from a secured Web browser running on a secured mobile device.

Consequently, we believe that cybersecurity can therefore in the future be addressed using explicit, computational OA representations that are attributed with both IP and cybersecurity obligations and rights.

Last, the inclusion of OSS or new CSS components within future OA C3CB software systems or AC will be amenable to current approaches to cybersecurity assurance, as we have outlined before (Scacchi and Alspaugh 2013d). Mission components can be assessed for cybersecurity characteristics, and assembled, without triggering reaccreditation. Similarly, evolutionary support for field deployed AC can allow rapid substitution of mission components that enable rapid, agile response to cybersecurity issues in mission components. However, legacy CSS components which were developed and deployed before current cybersecurity assurance challenges will need to rely on “air-gap” interfaces at deployment time that may be vulnerable to aggressive exploits delivered through mobile devices. **Figure 11** shows how the OA software system that includes mobile apps/widgets can be configured to employ encapsulation or containment vessels via virtual machines or virtual operating systems (outlined in the Figure) to provide a more cybersecure system architecture.

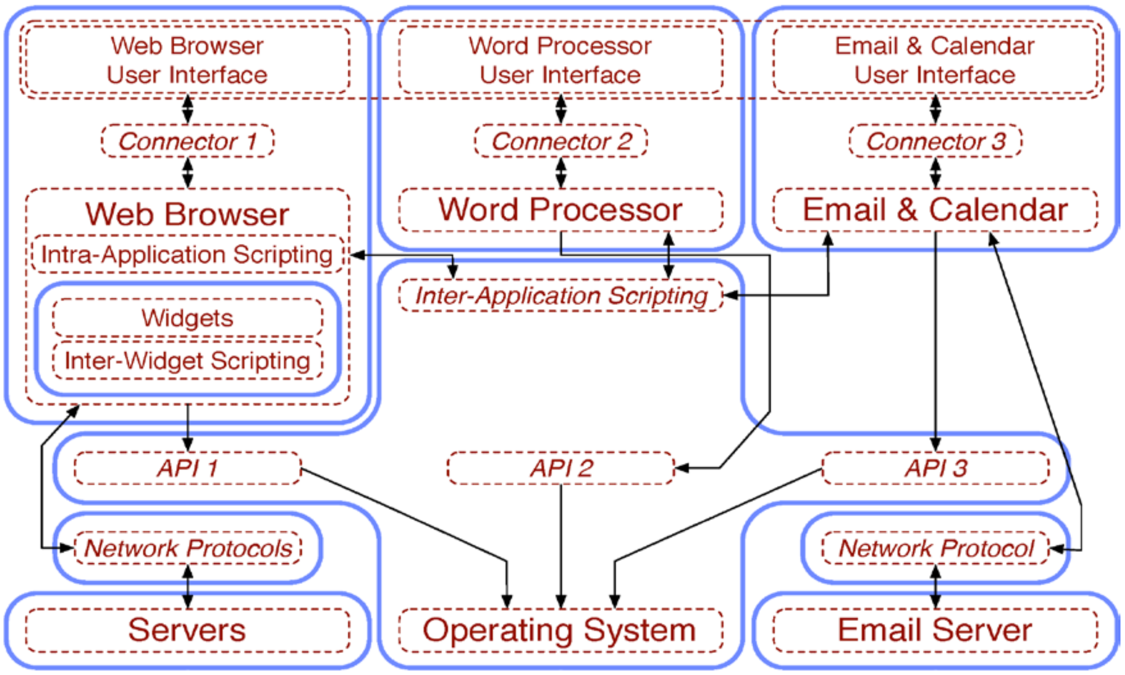


Figure 11. A configuration of security confinement vessels that encapsulate infrastructure software components and mission-specific widgets for the OA shown in Figures 8 and 9.

4. Software Component Build, Release, Deployment (BRD) Processes

C3CB applications represent complex software systems that are often challenging to produce, especially when conceived as bespoke systems. To no surprise, acquisition of these systems often requires a development life cycle approach, though some system elements may be fully-formed components that are operational as packaged software (e.g., commercial database management systems, Web browsers, Web servers, user interface development kits/frameworks). C3CB development is rarely clean sheet and less likely in the future. Subsequently, component-based system development approaches are expected to dominate, thus relegating system integrators (or even end-users) to perform any residual source code development, inter-app integration scripting, or intra-app extension script development. But software process challenges arise along the way (Scacchi and Alspaugh 2013b).

First, is again the issue noted earlier of whether there is an explicit, open source OA design representation, preferably one that is not just a diagram, but instead is expressed in an architectural design language. With only a diagram or less, then is little/no guidance for how to determine whether a resulting software implementation is verifiable or complaint with its OA requirements or acquisition policies, such as provision or utilization of standardized, open APIs, intended to increase software reuse, selection of components from alternative producers, or post-deployment end-user extensions (Kendall 2015).

Second, is the issue arising from system development practices based on utilization of software components, integrated sub-systems, or turnkey application packages. These software elements come with their own, possibly unknown requirements that are nonetheless believed to exist and be knowable with additional effort (Alspaugh and Scacchi 2013). They also come with either OSS code or CSS executables, along with their respective APIs. These components must be configured to align with the OA specification. Consequently, software tool chains or workflow automation pipelines are utilized to build and package internal/external executable, version-controlled software releases. We have found many diverse automated software process pipelines are used across and sometimes within software integration activities (Scacchi and Alspaugh 2013b). These pipelines take in OSS code files, dependent libraries, or repositories (e.g., GitHub), build executable version instances that are then subjected to automated testing regimes that include simple “smoke tests” and extensive regression testing. Successful builds that eventually turn into packaged releases that may or not be externally distributed and deployed as ready-to-install executables. While this all seems modest and tractable, when one sees the dozens of different OSS tools used in different combinations across different target platforms, then it becomes clear that what is simple in the small, is a complex SE activity when the scale of deployment increases.

Another complication that is now beginning to be recognized within and across BRD processes and process automation pipelines arises in determining when and how different BRD tool chain versions/configurations can mediate cybersecurity requirements in the target system being built. We have seen when software builds and deployed released are assumed to integrate to functionally equivalent CSS components, which are not included in releases, due to IP restrictions. We have also observed and reported how functionally equivalent variants as well as functionally similar versions may or may not be produced by BRD tool chains, either by choice or by unintentional consequence. This in our opinion gives rise for explicit open source models of BRD process automation pipelines that can be analyzed, tested, reused, and shared to determine whether release versions/variants can be verified and/or validated to produce equivalent or similar releases that preserve prior cybersecurity obligations and usage rights.

Last, mixing new OSS and CSS components with legacy apps wrapped within widgets will complicate build and release processes and obscure deployment processes (not to mobile devices that can operate new components). Legacy apps encapsulated within mission-specific widgets will commonly need to dynamically link executable binary components, which in turn increases the challenges in their testing and cybersecurity assurance, both during development and field deployment. In order to mitigate these technical challenges while enabling more agile software component system integration, multi-component OA configurations should explicitly declare pre/post conditions on acceptable input/output parameter values, along with exceptional values, that in turn can be independently verified or validated.

5. Software Component Evolution Practices Transmitted Across the OA Ecosystem

Software evolution is among the most-studied of SE processes. While formerly labeled as “software maintenance,” a profitable activity mediated through maintenance contracts from software producers to customers, the world of OSS development projects and practices suggest a transition to a world of continuous software development—one that foreshadows the emergence of continuous SE processes, or software life cycles that just keep cycling until interest falters or spins off into other projects. OSS development projects rely on OSS tools that themselves are subject to ongoing development, improvement, and extension, as are the software platforms, libraries, code-sharing repositories, and end-user applications utilized by OSS developers to support their development work. Developers entering, progressing, or migrating within/across OSS projects further diversifies the continuous development of the most successful and widely used OSS components/apps. This dynamism in turn produces many ways for how OSS systems, or OA systems that incorporate OSS components evolve.

Figure 12 portrays different software evolution patterns, paths, and practices we have observed arising with new C3CB applications (Scacchi and Alspaugh 2012). Here we see paths from a currently deployed, executable system release, to a new deployed release—something most of us now accept as routine as software updates are propagated across the Internet from producers, through integrators, to customers and end-users.

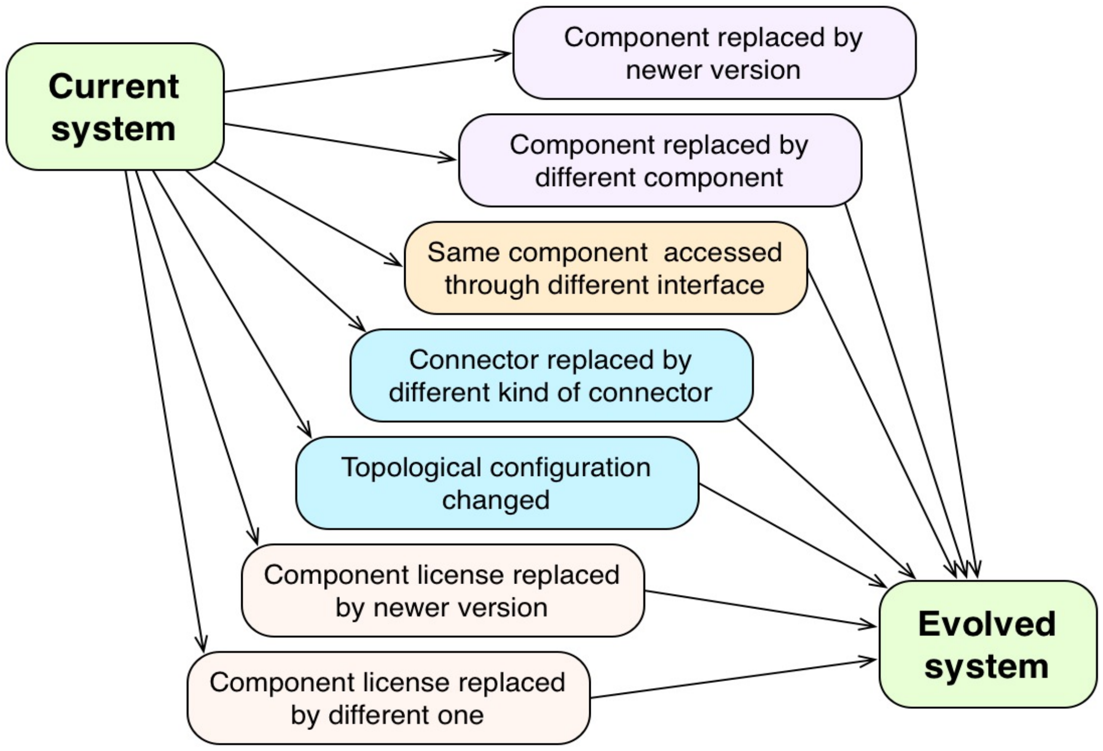

Current system
Component replaced by newer version
Component replaced by different component
Same component accessed through different interface
Connector replaced by different kind of connector
Topological configuration changed
Component license replaced by newer version
Component license replaced by different one
Evolved system

Figure 12. Different paths and mechanisms through which conventional and mobile OA software systems can evolve (Scacchi and Alspaugh 2012).

Integrated OA systems can evolve through upgrades of functionally equivalent component variants (patches) as well as through substitution of functionally similar software components sourced from other producers or integrators. In **Figure 13**, we show a generic situation that entails identifying how an OA consistent with that depicted in **Figure 8** may accommodate the substitution and replacement of a locally installed word processor application with a remote Web-based word processing software services (for example, *Google Docs* or *Microsoft Office 365*). This is capability is a result of utilizing an OA that constitutes a reference model aligned with a vendor-neutral software product line. This is also a capability sought by customer organizations, and sometimes encouraged by software producers to accommodate their evolving business models (discussed below). While the OA remains constant, the location of the component has moved from local to remote/virtual, as has its evolutionary path. Similarly, the cybersecurity of the local versus remote component has changed in ways that are unclear, and entail a different, evolved assurance scheme.

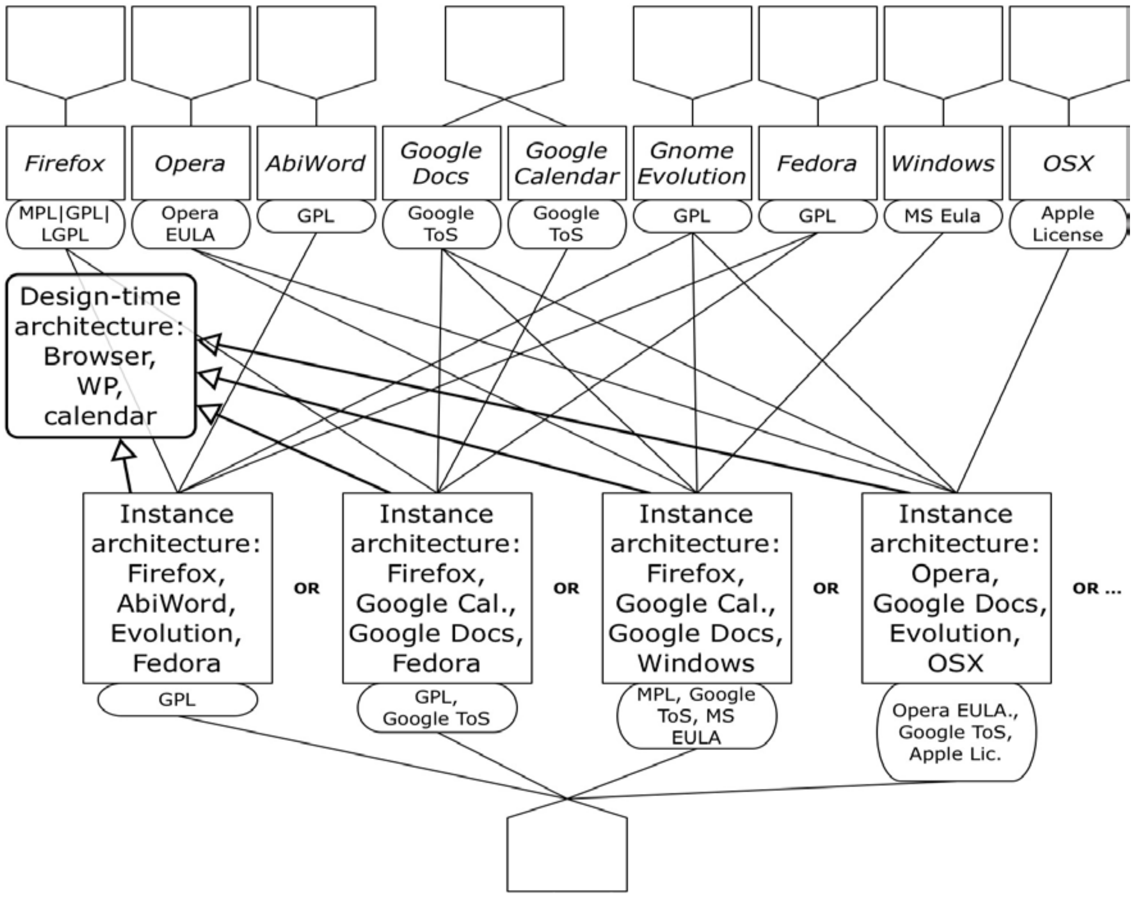

Firefox
Opera
AbiWord
Google Docs
Google Calendar
Gnome Evolution
Fedora
Windows
OSX
MPL|GPL|LGPL
Opera EULA
GPL
Google ToS
Google ToS
GPL
GPL
MS Eula
Apple License
Design-time architecture: Browser, WP, calendar
Instance architecture: Firefox, AbiWord, Evolution, Fedora
OR
Instance architecture: Firefox, Google Cal., Google Docs, Fedora
OR
Instance architecture: Firefox, Google Cal., Google Docs, Windows
OR
Instance architecture: Opera, Google Docs, Evolution, OSX
OR ...
GPL
GPL, Google ToS
MPL, Google ToS, MS EULA
Opera EULA., Google ToS, Apple Lic.

Figure 13. Alternative configurations of integrated instance releases of components consistent with the OA in Figure 2 that are treated as functionally equivalent by customer organizations (Scacchi and Alspaugh 2012).

Next, any common development technology used to support production or integration of mission components with shared infrastructure components must recognize that these technologies and components are all subject to independent, mostly autonomous evolution practices within the Defense Community. For example, OZP has been undergoing evolution, including its migration to Java 8 sourced by Oracle, where this move may disrupt the correct operation of widgets already produced using Java 7 common development technologies. Similarly, the many new OSS and CSS components seen in **Figure 3** will evolve due to practices arising in the competitive marketplace, while legacy mission components wrapped within widgets will have obscure or opaque evolution practices that are locked into legacy Defense Community component providers. Legacy components will also limit how their encapsulating widgets can evolve, potentially due to architectural mismatches or dependencies to legacy systems that are no longer supported, operational, or compatible with current platform technologies (Velasco-Elizondo, Dwivedi, Garlan et al. 2013).

Overall, the evolution of software components, component licenses, component interconnects and interconnections, and interconnected component or AC configurations are now issues that call for research efforts to help make such patterns, paths, and practices more transparent, tractable, manageable, and scalable within an OA software ecosystem, as well as for customer organizations that seek the benefits of openness, sharing, and reuse.

6. New Business Models for Acquisition of Software Components and Widgets

The last issue we address is the newest in this set of six for consideration for new acquisition research. While the field of acquisition research and practice has long paid attention to software economics, the challenges of software cost estimation are evolving in light of new business models being put into practice by software producers and system integrators. In the past, a single contractor responsible for both software production and system integration often managed software development projects. Costs could be assessed through augmentation to internal business accounting practices (e.g., budgeting, staffing workloads, time-sheet reports, project schedules, etc.). But a move to OA ecosystems means that multiple producers can participate, and OA schemes accommodate switching among providers, while a system is being integrated, deployed, or evolved in the field. This in turn coincides with new ways and means to electronically distribute software updates, components, or applications, as well as new ways to charge for software. OSS components may be acquired and distributed at “no cost,” but their integration and evolution charged as service subscription, or as time-effort billings.

We have already seen other alternatives for costing or charging for software that include: franchising; enterprise licensing; metered usage; advertising supported; subscription; free component, paid service/support fees; federation reciprocity for shared development; collaborative buying; donation; sponsorship; free/open source software (e.g., Government OSS – GOSS); and others. So how are customer organizations, especially in the Defense Community where software cost estimation practices are routine, suppose to estimate the development or sustaining costs of the software components or integrated systems they acquire and evolve, especially when an OA system allows for producers whose components come with different costing/billing schemes? This is an open problem for both acquisition research and software engineering practice.

Overall, new OSS and CSS components are experiencing a rapid diversification of acquisition cost models and practices, while legacy components are generally tied to single-source contractors which utilize legacy components as a cost-avoidance practice. All of the preceding five factors further obfuscate how to estimate or measure software component/AC development costs, schedules or time to delivery/usage. So acquisition costs of systems that mix and match new OSS and bespoke CSS components, together with legacy CSS components, will be difficult to cost estimate or cost manage. This in turn will limit the efficacy of BBP 3.0 practices for such systems.

Recommendations for Achieving Better Buying Power for Secure Mobile OA Software Systems with Widgets/Apps Across Diverse Acquisition Scenarios

Better Buying Power is part of DoD's mandate to do more without more by implementing best practices in acquisition [Kendall 2015]. BBP identifies seven areas of focus that group a larger set of a few dozen initiatives that offer the potential to restore affordability in defense procurement and improve defense industry productivity. One of the seven areas focuses on promoting or increasing competition, and this area includes an initiative to “enforce open system architectures and effectively manage technical data rights” [Defense Acquisition University 2012]. Technical data rights pertain to two categories of Intellectual Property (IP): they refer to the Government's rights to (a) *technical data* (TD – e.g., product design data, computer databases, and computer software documentation); and (b) *computer software* (CS – e.g., source code, executable code, design details, processes, and related materials). These rights are realized through IP licenses provided by system product or service providers (e.g., software producers) to the Government customer, so long as the customer fulfills the obligations stipulated in the license agreement (e.g., to indicate how many software users are authorized to use the licensed product or service according to a fee paid).

Addressing Better Buying Power Issues

As already noted, our acquisition research has focused on issues addressing conventional and now mobile OA systems and IP licenses since 2008 [Scacchi and Alspaugh 2008], as well as forward to the acquisition of secure OA systems for command and control (C2) and enterprise information systems [Scacchi and Alspaugh 2011, 2012b, 2013b], where security requirements can be expressed in a manner similar to IP obligations and rights. Therefore, we turn to identify how a sample of different goals of the BBP initiatives interact or relate to the trends and challenges examined so far in this paper. Representative BBP goals are highlighted, and then followed by a brief examination.

- *Increase competition* – One central purpose for acquiring OA systems is to increase the likelihood of competition among system producers who can provide software components that can be replaced by similar offerings by other component producers. We demonstrate how this can work when system architectures are explicitly modeled, and their software components and interconnections are similarly specified in an open manner [Alspaugh, Asuncion, and Scacchi 2013, Scacchi and Alspaugh 2012a].
- *Adopt OA systems that utilize standardized interfaces* – Open system architectures that can accommodate common components from alternative producers require that components utilize standardized interfaces, whether in the form of: open Application Program Interfaces (APIs); standard data exchange protocols; or standard data representations, formats, and meta-data [Scacchi and Alspaugh 2008]. But also noted earlier, app and widget components at present have a plethora of standardized interfaces, and it is unclear which will survive, be sustained, be widely adopted (inside/outside of DoD), and be evolved [Endres 2013a].
- *Utilize open source software components (including widgets and apps) where appropriate to reduce costs* – another aspect of openness that OA systems embrace and DoD policy accepts is to utilize system components developed as open source software (OSS) [DIS12]. Utilization of OSS components, along with composing OA systems that incorporate OSS and closed, proprietary components, requires careful attention to the management and analysis of multiple IP licenses that apply to different OA system components,

as well as determining what overall IP and/or cybersecurity rights and obligations apply to the overall system [Alspaugh, Asuncion, and Scacchi 2013, Scacchi and Alspaugh 2012a], especially for secure C2 systems [Alspaugh, Asuncion, and Scacchi 2013, Scacchi and Alspaugh 2013b, 2013c].

- *Increase small business roles and opportunities* – one way to increase competition in the realm of OA systems is to identify where smaller scale software applications (apps) or widgets can be utilized, which might be produced by small businesses or startup ventures which dominate much of the online markets for Web-based or mobile device apps/widgets. Small businesses may further be advantaged by their utilization of OSS infrastructure components, platforms, or remote services, since large commercial contractors may not see sufficient profit margins to develop proprietary alternatives. So OA systems that accommodate OSS components that can integrate custom apps/widgets into innovative C3CB system capabilities, may then realize new opportunities for DoD customers. Other small business opportunities may similarly arise for such ventures that focus on emerging cybersecurity assessment or tool development services.
- *Use technical development phase for true risk reduction and rapid prototyping* – In looking forward, there is potential interest in seeing the BPP initiative evolve to also address risk as an implicit cost driver. This might allow for innovative ways and means to reduce emerging risks through accelerated or "look ahead" system acquisition and development approaches that emphasize increased reliance on rapid prototyping. This kind of rapid prototyping might even be performed by appropriately trained end-users or warfighters. A move towards OA systems for Web-based and mobile devices that rely on apps/widgets retrieved from online marketplaces, that can be composed through interpretive software program "scripting" and mashup techniques, is a clear example of this [Endres 2013, George, Morris, Galdorisi, *et al.* 2013, Guertin and Womble 2012, Scacchi and Alspaugh 2013a]. Thus, it is not surprising to find such emerging techniques being investigated and assessed for possible production of new C3CB capabilities.
- *Doing more without more* – an overall summary of BBP initiatives is that of focusing attention for how to make acquisition more agile, to do more without more, and to develop a new generation acquisition workforce that can enact acquisition processes that are thin and flexible when needed, yet robust and cost-effective, while also being amenable to continuous improvement. This is indeed a real challenge to fulfill, and beyond the scope of what current acquisition practices are likely to achieve without targeted investment in acquisition improvement research. To be clear, one just needs to consider emerging opportunities (and potential asymmetric cybersecurity threats) that arise through the desire to develop next-generation C2SC that are to be composed from apps/widgets that can operate on Web-based/mobile devices. What are the best processes or practices for acquiring, developing, and sustaining deployed systems that are to be built using these new

software technologies (e.g., apps/widgets for mobile devices)? How should these processes and practices be adapted to accommodate personal devices (e.g., Apple iPhones, Android tablet, Microsoft Mobile Phone, Blackberry 10 phone) that individual warfighters, joint force troops, or contracted service providers bring with them into the battlespace? How must acquisition processes be best adapted to accommodate and rely on software supply chains that arise around consumer-oriented app marketplaces as possible ways/means for *doing more* (e.g., rapidly prototyping warfighter composable C2 app/widget mashups [George, Morris, Galdorisi, *et al.* 2013]) *without more* (e.g., warfighters who bring their own mobile computing devices for use in C3CB contexts) [George, Galdorisi, Morris, *et al.* 2014]? Once again, these are critical questions to address and resolve through new acquisition research and supporting technology development.

Improving and streamlining acquisition processes for secure OA systems

The transition to the development, deployment, and sustainment of software-intensive conventional or mobile systems based on an OA means that new or revised acquisition processes may be needed. In particular, we believe such advances call for (a) the adoption of open business models within DoD and its industry partners, (b) open source approaches to creating Web-based acquisition processes [Scacchi 2001] that specifically address BBP initiatives, and (c) employing techniques for streamlining these processes [Choi and Scacchi 2001, Nissen 1998, Scacchi and Noll 1997, Scacchi 2001] for secure OA systems. Each is described in turn in this section.

Encourage the adoption of acquisition business models in open source formats

One goal of BBP initiatives is to reduce costs by improving competition. This goal is independent of whether OA software systems include conventional or mobile software components. Such a situation may be disconcerting to legacy software producers or contractors who are long experienced with the long-term development of proprietary, large-scale software systems with closed architectures that are subject to traditional, cumbersome, and costly software product licenses and license management regimes [Anderson 2012, Konary 2009]. A move towards agile and adaptive development of secure OA systems based on software components, that can be developed/integrated more rapidly and at lower cost with more favorable IP licenses, represents a new acquisition strategy [Reed, Benito, Collens, *et al*. 2012, Scacchi and Alspaugh 2013b]. This suggests the need to incentivize software producers and system integrators, so as to insure their ability to effectively produce both proprietary and OSS components that are economically viable yet cost effective to the Government over the life of such systems. The overall BBP mandate recognizes this situation, but does not specify the means for how best to accomplish it. We believe one promising candidate is for Defense Enterprises and Program Offices to adopt new open business models.

The business models we have in mind should be rendered in an open source format. Such models should be computer-processable (i.e., amenable to automated enactment support) and transparent to participants in the acquisition workforce (e.g., available through Web-based application systems [Scacchi 2001, Scacchi and Noll 1997]). They should be similarly open to participants in software producer, system integrator, and customer enterprises. These models should incorporate a product line of common/reusable open system architectures that can integrate functionally similar software components in order to realize domain-specific system solutions (e.g., for domains like command and control, weapon systems, or enterprise computing) [Bergey and Jones 2010**,** Guertin and Clements 2010, Jones and Bergey 2011, Reed, Benito, Collens, *et al.* 2012, Scacchi and Alspaugh 2012b**,** Software Engineering Institute 2007**,** Womble, Schmidt, Arendt Fain 2011]. These business models could incorporate Web-based computational models of acquisition processes [Nissen 1998, Scacchi and Noll 1997, Scacchi 2001] that can track the system development and support processes that surround the OA product line system models. Finally, these business models should highlight which acquisition or system development processes, or OA system features, require attention to IP licenses.

Prior research has demonstrated that significant cost reductions and process streamlining are possible when open source business process models are utilized [Choi and Scacchi 2001, Nissen 1998, Scacchi and Noll 1997, Scacchi 2001]. These kinds of models can be subjected to performance measurement across multiple acquisition process enactments, continuous improvement, and process redesign by the acquisition workforce [Scacchi 2001]. Now we propose to enhance and extend their value through the incorporation of OA system models. While demonstrating such a capability is beyond the scope of this study, prior research results suggest the plausibility of such an approach. So future acquisition research targeting BBP may investigate the creation of open business models that can be openly accessed, reused, modified, and redistributed where appropriate.

Encourage the development, (re)use, and refinement of open source models of acquisition processes

As noted, prior research has demonstrated the value and real payoffs of Web-based computational models for Defense acquisition processes. However, many technological advances, organizational transformations, and shifting Defense priorities have occurred since these results were first demonstrated and deployed years ago. Our own studies on design of secure OA system product lines with conventional or mobile components are an example of technological advances not addressed in our earlier process models. But without explicit, open source process models that can be enacted through Web-based user interfaces (i.e., Web browsers accessing remote application services while tracking process enactment progress and performance parameters), then the ability to realize their benefits like process streamlining and cost reduction are elusive and difficult to manifest. Among the reasons for why this is so includes overcoming gaps for how best to: (a) monitor and measure acquisition process performance without automated enactment support; (b) redesign legacy processes to better accommodate technical advances and to remove ineffective bureaucratic procedures, or that transform acquisition processes in ways that do more with less while also empowering the acquisition workforce; (c) design new acquisition processes like those for acquiring secure, component-based OA software systems subject to multiple IP licenses; and (d) accommodate software IP licenses and license management regimes as acquisition process cost elements. To better understand what gaps exist in these four areas, we now describe techniques for streamlining the acquisition processes for secure OA system.

Develop and employ techniques for streamlining acquisition processes for secure OA systems

A goal of this paper is to identify ways and means for streamlining acquisition processes for secure OA systems. In particular, we focus on four kinds of techniques that can be used to streamline such processes in ways that are responsive to the BBP initiative for open system architectures subject to complex IP licenses. These techniques are illustrative rather than exhaustive, as other kinds of techniques in other areas are also expected to exist and be available for practice by the acquisition workforce.

Acquisition Process Measurement and Assessment – The most direct way to determine the efficiency and effectiveness of acquisition processes is by measuring their structural attributes. Such measures indicate acquisition process metrics such as (a) length of longest path of process steps/actions (process length); (b) number of distinct process paths (process width); (c) number of sub-process levels (process depth); (d) total number of process steps (process size); and (e) process size divided by process length (process parallelism), and others metrics [Nissen 1998]. But without an explicit, open source, graph-based model of acquisition processes, such measurements are impractical or implausible. Nonetheless, such metrics are a key for where to look for process improvement or process redesign opportunities. One might also recognize that not explicitly accounting for where software licenses are negotiated or license trade-off analysis done underspecifies some acquisition processes, for example. Similarly, as OA systems may include software components subject to different licenses [Alspaugh, Scacchi, and Asuncion 2010], then how are component-component license interactions assessed or analyzed, if at all? If acquisition processes do not explicitly account for new acquisition or license management activities that emerge due to advances in OA system development, then such processes are underspecified, which means their costs are hidden and difficult to control/minimize. Thus, if the goal of BBP is to help improve the affordability of OA systems within the DoD, then we need to be able to systematically model, measure, and assess our acquisition processes [Scacchi 2001]. Similarly, we need to better understand how to measure and assess open business models for use within DoD and its industry partners to incentivize and continuously improve competition and Defense affordability.

Acquisition Process Redesign and Evolution – Once we have the ability to measure and assess current/emerging acquisition processes for secure component-based OA systems; we can analyze (or simulate) them in ways that reveal process redesign opportunities and transformation heuristics [Choi and Scacchi 2001, Nissen 1998, Scacchi and Noll 1997, Scacchi 2001]. Among the acquisition process pathologies we seek to identify are those where measured processes reveal sub-processes with *low effectiveness* (indicating high levels of iterative rework), *low efficiency* (indicating slow or bureaucratically cumbersome process steps that add little marginal value to process completion), and *problematic sub-processes* (indicating underspecified process steps, steps that generate processing delays due to missing/or incorrect acquisition data, or inappropriate automated process enactment support). For example, current processes that assume long-term acquisition of monolithic software systems with proprietary components integrated within a closed architecture, are not well-suited to address the challenges for acquiring secure OA systems that integrate software components from different online repositories. We also place the acquisition workforce at a disadvantage if we do not empower them with the ability to measure, assess, and adaptively redesign their processes as technological advances like component-based OA systems are to be acquired. New software component technologies and software ecosystem niches [Scacchi and Alspaugh 2012a] are also emerging which necessitate new *continuous development processes* and new license management practices, and thus redesign/evolution of acquisition processes [Scacchi and Alspaugh 2013a]. These examples all point to new opportunities to redesign, evolve or other transform existing acquisition processes to better fit the challenges posed by the development, deployment, and support of secure, component-based OA systems. Finally, we can empower the acquisition workforce to realize continuously improved acquisition processes if we can provide them with the training and resources for modeling, analyzing, and redesigning their acquisition processes in ways that empower them to utilize Web-based automated process enactment systems, which also allow them to try out and walkthrough alternative process redesigns before committing to their use in daily operations.

Design New Acquisition Processes – Across the DoD community, there are many variations in practice for how to specify and model the architecture of a software-intensive system. Some practices focus attention primarily on identification of major components or abstract layers, while minimizing (or ignoring) attention to interfaces and interconnections, which are more challenging to identify, manage or standardize. However, the BBP initiative for OA systems points to the need for managing explicit interface specifications that identify and reinforce the use of standard interfaces [Defense Acquisition University 2012]. Without such interface and interconnection specifications, it is not possible to determine the scope or potential conflicts/matches between the IP licenses (and thus TD rights) for the overall system architecture. In contrast, prior research has demonstrated that component-based OA systems become tractable and evolvable from IP license management and security perspectives when the system architecture of components, connectors, and interfaces are explicitly modeled using contemporary architecture description languages or similar means [Alspaugh, Scacchi, and Asuncion 2010, Scacchi and Alspaugh 2011, 2012a, 2012b, 2013b].

The use of standard interfaces allows for simpler renderings of OA system structure, and thus simplifies license analysis. Further, once interfaces and interconnections become explicit, software component producers, system integrators, and/or system customers can determine/negotiate which interfaces should be standardized in order to improve competition and affordability. These standards may then define acceptable data types, relationships between data types, data attribute value ranges, and exceptional data values in ways that are open, sharable, and reusable, as well as extensible when appropriate. Such improvements become possible by enabling an agile, adaptive ecosystem for software components of different size and capability relative to OA system product lines for different application domains [Reed, Benito, Collens, *et al.* 2012, Scacchi and Alspaugh 2012a, 2013b]. Therefore, another important technique for streamlining the acquisition of secure, component-based OA systems, in line with BBP initiatives, is to provide the acquisition workforce with the resources and automated support to design and computationally enact new acquisition processes (i.e., explicitly modeled processes [Choi and Scacchi 2001, Nissen 1998, Scacchi and Noll 1997, Scacchi 2001]), where the processes are open, agile, and adaptive. Such modeled processes may also then be shared, reused, continuously improved, and redistributed across the ecosystem of Defense Enterprises and Program Offices [Scacchi and Alspaugh 2016].

Cost Management as an Acquisition Process Design Element – Part of the promise of the move to OA systems stems from their perceived potential to reduce acquisition life cycle costs, improve competition, and improve Defense affordability [Defense Acquisition University 2012]. But where and how are the associated cost factors or cost drivers for OA systems identified, tracked, and managed? After all, if we do not know where the cost factors are, or what activities, conditions, or events drive OA system acquisition costs, and then we cannot effectively control such costs, nor make well-informed system capability/cost tradeoffs. For example, people who manage the acquisition of large-scale software systems within various Defense Enterprises are familiar with the many types of end-user license agreements for proprietary, closed source software systems [Anderson 2012]. In contrast, these people may not know how best to manage the acquisition of OA systems whose software components are jointly subject to different OSS or proprietary licenses.

The acquisition workforce has also learned in practice that software IP licenses are subject to change over time. However, one consequence is that long-lived or widely used software systems become more costly and much less amenable to technology substitution or vendor replacement, thereby reducing competition due to vendor lock-in. This works against Defense affordability. In contrast, emerging online repositories offer different kinds of software components with different functional capabilities (described earlier), along with different IP licenses and end-user licenses (e.g., low cost, per user licenses). These repositories of software components represent a means for increased competition and affordability, but subject to different acquisition, development or integration processes that are just coming to light. Accordingly, we believe that streamlining the acquisition process for secure, component-based OA systems requires that IP license cost obligations (e.g., license fees for end-user agreements) and license management regimes need to be incorporated into: process measurement and assessment, process redesign and evolution, and design of new acquisition processes. This is also a subject for further acquisition research, but one offering practical near-term consequence.

Discussion and Conclusions

The trends, concerns, and recommendations identified above point to substantial challenges in identifying what can be done to both realize cost-effective BBP for Web-based and mobile device software apps, and to do so in ways that enable and empower the acquisition workforce in the years ahead. Technology, better buying practices, new business models, and new cybersecurity requirements all point to the need for future research and development of new acquisition support technologies, work processes, and guidance practices. The goal is to make sure that acquisition time and effort does not become the main cost and the main risk factor going forward on the path to agile OA Web-based or mobile compatible C3CB system development, deployment, and sustaining system evolution.

At this point, we see at least four key areas of opportunity for *future acquisition research* for both conventional and mobile OA software systems intended for application domains like C3CB.

First, we need to develop **worked examples** of well-formed OA system architectures that are appropriate for C3CB system capabilities, and that accommodate Web-based apps, widgets, and mobile devices. Such OA system architectures should specify representative and standardized component interfaces. The examples should also include carefully specified shared agreements that account for different IP licenses and diverse business models of software producers, system integrators, and multiple end-user organizations who must collectively act in ways that enable agile development and adaptive evolution of demonstrable C3CB system capabilities.

Second, we need robust **open source models** of application security processes and reusable cybersecurity requirements that account for exigencies in heterogeneous app/widget software ecosystems, account for software evolution dynamics, formation and continuous improvement of automation-compatible shared agreements, and more. These models should account for description of current process practices, prescription of required verification and validation activities and outcome (deliverable documents or online artifacts), and proscription of what tools/techniques to use, by whom, when, where, and how.

Third, we need reasonably precise, human readable and computer processable **domain specific languages** (DSLs) for specifying OA systems and their corresponding IP licenses and cybersecurity requirements, along with automated analysis capabilities within a software obligation and rights management system (SORMS). The purpose for the SORMS is to provide automated support for continuously assessing and continuously improving cybersecurity and IP license requirements for dynamically evolving Web/mobile C3CB system-based capabilities. The DSLs needed must be able to specify and operationalize the shared agreements between different DoD organizations, government agencies, and commercial enterprises involved in producing, integrating, or evolving component-based OA C3CB system capabilities. Relying instead on informal natural language documents or agreements that require ongoing legal review is merely a way to insure ambiguities, inconsistencies, and mistaken understandings that contribute to the loss of cost control, reduce acquisition effectiveness and unnecessarily prolong acquisition cycle times.

Fourth, technological advances can and will stimulate innovation in OA software system components, widgets or apps across the software supply chain network suggested in Figure 1. However, we note that six issues in OA software system development, deployment and evolution will increasingly create diverse acquisition scenarios for both conventional and mobile OA systems, especially those incorporating Web and mobile software components via apps and widgets. In particular, this suggest that new fundamental challenges will arise in determining how the acquisition workforce along with software component producers, system integrators, and customer organizations will need to continuously assure the cybersecurity of the resulting OA systems as they develop and evolve. This suggests to us the need for the acquisition research community, along with other research agencies, to look for new ways and means to stimulate innovation in cybersecurity principles and practices that can better accommodate the other technical challenges/issues now arising with mobile OA software systems.

Specifically, we believe there is much promise to follow from investigations in how technologies and techniques associated with blockchains and smart contracts may be utilized as ways and means for modeling and analyzing OA software supply chains, and to associate such models with explicit OA system model representations, along with the IP licenses and cybersecurity requirements associated for both conventional and mobile OA software systems and their software components.

Our study reported in this paper identifies a set of both broad business practices; technical issues and software supply chain risks that can dilute the cost-effectiveness of Better Buying Power efforts. It similarly suggests that current acquisition practices aligned with BBP can also give rise to acquisition management activities that can dominate and overwhelm the costs of OA system development. This adverse condition can arise through app/widget vetting, new software business models, opaque and/or underspecified acquisition management processes, and the evolving interactions of new software development and deployment techniques. Unless proactive investment in acquisition research and development can give rise to worked examples, open source models, and new acquisition management system technologies, the likelihood of acquisition management dominating agile development and adaptive deployment of component-based OA C3CB system capabilities is remote.

Our research identified and analyzed how new software component technologies like OSS infrastructure components, common development technology components, and mission-specific widgets for Web-based and/or mobile devices, along with their intellectual property (IP) license and cybersecurity requirements, engineering and evolution processes, and cost estimating practices interact to drive down (or drive up) total system costs across the system acquisition life cycle. The availability of such new scientific knowledge and technological practices can give rise to more effective expenditures of public funds and improve the effectiveness of future software-intensive systems used in government and industry. Thus, a goal of this paper was to explore new ways and means for achieving cost-sensitive acquisition of OA software systems, as well as identifying factors that can further decrease or increase the costs of such systems.

We identified and examined six areas for research arising at the intersection of software engineering and acquisition that now confront the Defense Community (and perhaps other industries as well). These six issues areas include: (1) the lack of architecture representations and schemes for discovering or specifying OA system designs; (2) OA systems that integrate components or applications subject to diverse, heterogeneous IP licenses; (3) how to manage the cybersecurity of OA systems during system design, development, and deployment; (4) software process challenges and evolving disruptions in seemingly mundane process automation pipelines; (5) software evolution patterns, path, and practices in OA ecosystems; and (6) how new business models are upending software cost estimation practices and outcomes. All of these research areas are readily approachable, and research results are likely to have significant practical value, both within the Defense Community and beyond.

These issue areas were investigated and addressed in the domain of command, control, communication, cyber and business systems (C3CB). We believe all are tractable, yet dense and sufficient for both deep sustained research study, as well as for applied research in search of near-term to mid-term practical results.

Last, the ultimate purpose for these worked examples, open source models, and domain-specific languages is to provide new training materials and reusable media that can be tailored and adapted over time to benefit the acquisition workforce, as well as for system producers, integrators, and customer enterprises.

In summary, what we call for is similar in kind to what we have already produced and applied in other software development domains, using then current technologies [Jensen and Scacchi 2005, Scacchi and Alspaugh 2008]. What we now call for is a reinvention and repurposing of these concepts, but in contemporary forms scaled and secured in ways that best meet the needs of the DoD program offices, acquisition program managers, and others in the acquisition workforce to best support BBP 3.0 initiatives for Web-based and mobile device software components (widgets, apps, plug-ins) [Scacchi ad Alspaugh 2015].

In related work (Scacchi and Alspaugh 2015), we have called for specific R&D investments into the development of open source, domain-specific languages for specifying open architecture representations (or architectural description languages) that are formalizable and computational, as well as supporting annotations for software license obligations and rights. While ADLs have been explored in the SE research community, the challenges of how software architectures mediate software component licenses and cyber security requirements are an open issue, with practical consequence. Similar, ADL annotations that assign costs or cost models in line with new software business models are an open problem area. We have also called for R&D investment in new SE tools or support environments which purpose is to provide automated analysis and support of OA systems IP and cybersecurity obligations and rights, as new requirements for industrial practice in large-scale software acquisition, design, development, deployment, and evolution. Such environments are the automated tools that could be used to model, specify, and analyze dynamically configurable, component-based OA software systems expressed using the open source architectural representation schemes or ADLs noted here. We believe these recommendations still merit attention and commitment of research funding.

Our research identifies and analyzes how OA CBC3 system capabilities can utilize software components and mission-specific widgets, with diverse IP license and cybersecurity requirements, and new software business models can interact to affect total system costs across the system acquisition life cycle. The availability of such new scientific knowledge and technological practices can give rise to more effective expenditures of public funds and improve the effectiveness of future software-intensive systems used in Defense Community, as well as elsewhere within government and industry. Hopefully, this paper serves to help throw light into how software engineering and acquisition research can inform and add benefit to software practices within the Defense Community through ways and means that further advance Better Buying Power opportunities and outcomes.

THIS PAGE INTENTIONALLY LEFT BLANK

References

Agre, J.R., Gordon, K.D., and Vasiliou, M.S. (2014). Practical Considerations for Use of Mobile Apps at the Tactical Edge, *Proc. 19th Intern. Command and Control Research and Technology Symposium* (ICCRTS), Paper- 035, Fairfax, VA, June 2014.

Alspaugh, T.A. And Scacchi, W. (2012). Security Licensing, *Proc. Fifth Intern. Workshop on Requirements Engineering and Law*, 25-28, September 2012.

Alspaugh, T.A. And Scacchi, W. (2013). Ongoing Software Development Without Classical Requirements, (with T. Alspaugh), *Proc. 21st. IEEE Intern. Conf. Requirements Engineering*, Rio de Janeiro, Brazil, 165-174.

Alspaugh, T.A, Scacchi, W., and Asuncion, H. (2010). Software Licenses in Context: The Challenge of Heterogeneously Licensed Systems, *J. Assoc. Info. Systems*, 11(11), 730-755.

Anderson, S. (2012). Software Licensing – Smart Spending in These Changing Times, *CHIPS: The Department of the Navy's Information Technology Magazine*, July-September, 28-31.

Bergey, J., & Jones, L. (2010). Exploring acquisition strategies for adopting a software product line approach. *Proc. 7th Acquisition Research Symposium*. Vol. 1, 111- 122, Naval Postgraduate School, Monterey, CA.

Bass, L., Clements, P., and Kazman, R., (2003). *Software Architecture in Practice*, 2nd Edition, Addison-Wesley Professional, New York.

Bollinger, T., (2003). *Use of Free and Open-Source Software (FOSS) in the U.S. Department of Defense*, The MITRE Corporation, 2 January. Accessed March 2016.

Choi, S.C. & Scacchi, W. (1990). Extracting and Restructuring the Design of Large Systems, *IEEE Software*, 7(1), 66-71.

Chudnovsky, O., Fischer, C. Gaedke, M. and Pietschmann (2013). Inter- Widget Communication by Demonstration in User Interface Mashups. *Web Engineering*, Springer-Verlag, Lecture Notes in Computer Science, Vol. 7977, 502-505.

Conley, K., Brockman, B., Diercks, P., George, A., Lam, W., Lozano, A., Palnter, R. and Tolentino, G. (2014). Achieving Information Dominance: Unleashing the Ozone Widget Framework. *Proc. 19th Intern. Conf. Command and Control Research & Technology Symposium* (ICCRTS), Arlington, VA, paper 109. June.

Defense Acquisition Guidebook (2014). *CVE--Common Vulnerabilities and Exposures*. Chapter 13.7.3.1.4, accessed April 2015. https://acc.dau.mil/CommunityBrowser.aspx?id=492079#13.7.3.1.4

Defense Acquisition University (2012). *Open Systems Architecture and Technical Data Rights...Management Approaches*, http://bbp.dau.mil/docs/Open %20Systems%20Architecture%20and%20Technical%20Data%20Rights %20.%20.%20.%20Management%20Approaches.pdf , accessed 30 October 2012.

Department of Defense (2012). *Joint Operational Access Concept*, Version 1.0, 17 January 2012, http://www.defense.gov/pubs/pdfs/JOAC_Jan%202012_Signed.pdf

Endres-Niggemeyer, B. (2013). The Mashup Ecosystem, in *Semantic Mashups: Intelligence Reuse of Web Resources*, Springer, 1-50.

Endres-Niggemeyer, B. (2013a). Mashups Live on Standards, in *Semantic Mashups: Intelligence Reuse of Web Resources*, Springer, 51-89.

Garlan, D., Dwivedi, V., Ruchkin, I. and Schmerl, B. (2012). Foundations and Tools for End-User Architecting. In D. Garlan and R Calinescu (Eds.), *Large-Scale Complex IT Systems. Development, Operation and Management*, Lecture Notes in Computer Science, Springer, vol. 7539, 157-182.

George, A., Galdorisi, G., Morris, M., and O'Neil, M. (2014). DoD Application Store: Enabling C2 Agility, *Proc. 19th Intern. Command and Control Research and Technology Symposium*, Paper-104, Alexandria, VA, June 2014.

George, A., Morris, M., Galdorisi, G., Raney, C., Bowers, A., and Yetman, C. (2013) Mission Composable C3 in DIL Information Environments using Widgets and App Stores. *Proc. 18th Intern. Command and Control Research and Technology Symposium*, Paper-036, Alexandria, VA, June 2013.

George, A., Morris, M. and O'Neil, M. (2014). Pushing a Big Rock Up a Steep Hill: Lessons Learned from DoD Applications Storefront, *Proc. 11th Annual Acquisition Research Symposium*, Vol. 1, 306-317, Naval Postgraduate School, Monterey, CA.

Guertin, N.H., Sweeney, R., and Schmidt, D.C. (2015). How the Navy Can Use Open Systems Architecture to Revolutionize Capability Acquisition: The Naval OSA Strategy Can Yield Multiple Benefits. *Proc 12th Annual Acquisition Research Symposium,* Monterey, CA, NPS-AM-15-004, May 2015.

Guertin, N. and Clements, P. (2010). Comparing Acquisition Strategies: Open Architecture versus Product Lines, Vol. 1, 78-90, *Proc. 7th Acquisition Research Symposium*, Naval Postgraduate School, Monterey, CA.

Guertin, N. and Womble, B. (2012). Competition and the DoD Marketplace, *Proc. 9th Acquisition Research Symposium*. Vol. 1, 76-82, Naval Postgraduate School, Monterey, CA.

Hanf, D. (2013). *MPE/AAE Business Model Framework Overview*. Mitre Corporation, personal communication, July 2013.

Jensen, C. and Scacchi, W. (2005). Process Modeling Across the Web Information Infrastructure, *Software Process Improvement and Practice,* 10(3), 255-272, July-September 2005.

Jones, L. and Bergey, J. (2011). An Architecture-Centric Approach for Acquiring Software-Reliant Systems, *Proc. 8th Acquisition Research Symposium*, Vol. 1, 32-49, NavalPostgraduate School, Monterey, CA.

Kazman, R. and Carriere, S.J. (1998). Playing Detective: Reconstructing Software Architecture from Available Evidence. *J. of Automated Sofware Eng.*, 6(2), 107-138.

Kendall, F. (2015). *Implementation Directive for Better Buying Power 3.0,* 9 April 15.

Konary, A. (2009). *Software Licensing and Entitlement Management: The Next Software Licensing and Entitlement Management: The Next Generation*, IDC White Paper, October 2009. http://learn.flexerasoftware.com/content/ECMWPSoftwareLicensingEntitlementManagement, accessed 2 February 2013.

Meyers, B.C. and Obendorf, P., (2001). *Managing Software Acquisition: Open Systems and COTS Products*, Addison-Wesley, New York.

Nissen, M.E. (1998). Redesigning Reengineering through Measurement-Driven Inference, *MIS Quarterly*, 22(4). 509-534

Reed, H., Benito, P., Collens, J., and Stein, F. (2012). Supporting Agile C2 with an Agile and Adaptive IT Ecosystem, *17th Intern. Command and Control Research and Technology Symposium* (ICCRTS), Paper-044, Fairfax, VA, June 2012.

Reed, H., Nankervis, J., Cochran, J., Parekh, R., and Stein, F. (2014). Agile, Adaptive IT Ecosystem: Results, Outlook, and Recommendations, *Proc. 19th Intern. Command and Control Research and Technology Symposium* (ICCRTS), Paper-011, Arlington, VA, June.

Scacchi, W. (2002). Understanding the Requirements for Developing Open Source Software Systems, *IEE Proceedings—Software*, 149(1), 24-39. Also see Scacchi, W. (2009). Understanding Requirements for Open Source Software, in K, Lyytinen, P. Loucopoulos, J. Mylopoulos, and W. Robinson (eds.), *Design Requirements Engineering: A Ten-Year Perspective*, LNBIP 14, Springer-Verlag, 467-494.

Scacchi, W. (2010). The Future of Research in Free/Open Source Software Development, in *Proc. ACM Workshop on the Future of Software Engineering Research* (FoSER), Santa Fe, NM, 315-319.

Scacchi, W. and Alspaugh, T. (2012) Understanding the Role of Licenses and Evolution in Open Architecture Software Ecosystems, *J. Systems and Software*, 85(7), 1479-1494, July.

Scacchi, W. and Alspaugh, T., (2013a). Streamlining the Process of Acquiring Secure Open Architecture Software Systems, *Proc 10th Annual Acquisition Research Symposium,* Monterey, CA, 608-623, May 2013.

Scacchi, W. and Alspaugh, T. (2013b). Processes in Securing Open Architecture Software Systems, *Proc. 2013 Intern. Conf. Software and System Processes*, 126-135, San Francisco, CA, May 2013.

Scacchi, W. and Alspaugh, T. (2013c). Challenges in the Development and Evolution of Secure Open Architecture Command and Control Systems, *Proc. 18th Intern. Command and Control Research and Technology Symposium*, Paper-098, Alexandria, VA, June 2013.

Scacchi, W. and Alspaugh, T.A. (2013d). Advances in the Acquisition of Secure Systems Based on Open Architectures, in *J. Cybersecurity & Information Systems*, 1(2), 2-16.

Scacchi, W. and Alspaugh, T. (2015). Achieving Better Buying Power through Acquisition of Open Architecture Software Systems for Web and Mobile Devices. *Proc 12th Annual Acquisition Research Symposium,* Monterey, CA, NPS-AM-15-005, May.

Scachi , W. and Alspaugh, T. (2016). Issues in the Development and Maintenance of Open Architecture Software Systems, *CrossTalk: The Journal of Defense Software Engineering*, (submitted 2016, accepted to appear in 2017).

Scacchi, W. and Noll. J. (1997). Process-Driven Intranets: Life Cycle Support for Process Reengineering, *IEEE Internet Computing*, 1(5):42-49.Northrop, L., & Clements, *et al*., Software Engineering Institute (2007). *A Framework for Software Product Line Practice*, Version 5.0.http://www.sei.cmu.edu/productlines/ frame_report/index.html

Velasco-Elizondo, P. Dwivedi, V. Garlan, D. Schmerl, B. and Fernandes, J.M. (2013). Resolving data mismatches in end-user compositions. *End-user development*. Springer Berlin Heidelberg, 2013. 120-136.

Weir, M. (2014). BYOD Topic: How Complicated Can Calendars Be? *J. Cybersecurity and Information Systems*, 2(1), 18-19.

Wikipedia (2016). *Software Architecture*, accessed March 2016. https://en.wikipedia.org/wiki/Software_architecture

Womble, B., Schmidt, W., Arendt, M., and Fain, T. (2011). Delivering Savings with Open Architecture and Product Lines, *Proc. 8th Acquisition Research Symposium*, Vol. 1, 8-13, Naval Postgraduate School, Monterey, CA.