UMD-AM-21-247

ACQUISITION RESEARCH PROGRAM
SPONSORED REPORT SERIES

Understanding and Modeling the Life-Cycle Cost Tradeoffs Associated with the Procurement of Open Systems

27 September 2021

Shao-Peng Chen, Research Associate

Peter Sandborn, Professor

William Lucyshyn, Research Professor and Director of Research

University of Maryland

Approved for public release; distribution is unlimited.

Prepared for the Naval Postgraduate School, Monterey, CA 93943.

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.

Abstract

Openness (of a system or architecture), though intuitively understood, remains difficult to quantify in terms of its value. Although commonly associated with cost avoidance, system openness can also increase costs. Previous efforts have relied on highly qualitative system analyses, with the results often articulated as an intangible “openness score”, for determining which of multiple system implementations is more open. Such approaches do not provide enough information to make a business case or understand the conditions under which life-cycle cost avoidance can be maximized (or whether there even is cost avoidance). This report presents a multivariate model that quantifies the relationship between system openness and life-cycle cost. A case study that evaluates the Acoustic Rapid COTS Insertion (A-RCI) Sonar System is provided.

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgement

This material is based in part upon work supported by the Naval Postgraduate School Acquisition Research Program under Grant No. HQ00341910006 and the Office of the Secretary of Defense. The views expressed in this report do not necessarily reflect the official policies of the Naval Postgraduate School nor does mention of trade name, commercial practices, or organizations

THIS PAGE INTENTIONALLY LEFT BLANK

UMD-AM-21-247

ACQUISITION RESEARCH PROGRAM SPONSORED REPORT SERIES

Understanding and Modeling the Life-Cycle Cost Tradeoffs Associated with the Procurement of Open Systems

27 September 2021

Shao-Peng Chen, Research Associate

Peter Sandborn, Professor

William Lucyshyn, Research Professor and Director of Research

University of Maryland

THIS PAGE LEFT INTENTIONALLY BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

Introduction

The United States faces several long-term budgetary challenges. The rising costs of mandatory entitlement programs, coupled with the budget deficits projected into the foreseeable future, create inevitable downward pressure on future DoD budgets. As an example, DoD's 2020 budget request for acquisition was $247 billion (about one-third of its total budget request): $143 billion for procurement and $104 billion for RDT&E. This amount is approximately 1 percent less than the amount appropriated for 2019, when adjusted for inflation. Under the 2020 FYDP, this downward trend is expected to continue through 2024, decreasing the budget to $230 billion adjusted for inflation. This seven percent decrease will constrain the funds available for recapitalization, modernization, and transformation of the military (CBO, 2019). Future DoD budgets will require hard decisions and force a reengineering of processes and the most efficient use of resources.

DoD weapon systems have historically been developed using acquired proprietary systems and interfaces. These make it challenging to modernize and reduce opportunities for competition. For example, the Air Force's program to upgrade the B-2 bomber's communications, networking, and defensive management will cost over $2 billion, since the prime contractor owns all the necessary proprietary technical data and software. Competing this effort was not a viable financial option (GAO, 2014).

A variety of strategies are being explored or reemphasized to increase the efficiencies of acquisition processes. One way for the DoD to minimize the cost and time needed to modify or upgrade weapon systems is by using a modular open systems approach (MOSA) for system design and development. When used appropriately, MOSA provides a degree of flexibility, enabling the integration of rapidly changing technologies. However, as with all approaches, there are costs as well as benefits. This report explores a business case methodology to assess the cost effectiveness of MOSA.

Modular Open Systems Approach (MOSA)

System openness refers to the extent to which system components (e.g., hardware and software) can be independently integrated, removed, or replaced without adverse impact on the existing system. Current DoD policy calls for the use of modular open systems design to the maximum extent possible. In fact, the acquisition strategy for a given system must identify where, why, and how modular open systems will be used (DoD, 2015). Conventional wisdom supports the notion of open systems, but quantifying the actual cost avoidance remains elusive. The objective of this report is to quantify the relationship between system openness and life-cycle cost.

Open Systems Architecture, now referred to as Modular Open Systems Approach (MOSA), has been widely endorsed by the Department of Defense since the mid-1990s. MOSA promotes the use of modular design to encourage companies to improve and manufacture technology that is interoperable with the DoD's current system. Formally, the DoD defines MOSA as "a technical and business strategy for designing an affordable and adaptable system." An example of this could be the use of radar technology on an aircraft. Under MOSA, the radar technology could be replaced or upgraded without replacing the whole aircraft or numerous related subsystems. Closed systems architecture, on the other hand, effectively restricts access to configuration and programming information from outside parties. Closed systems often make upgrading a piece of equipment difficult and costly. Further, these closed systems can lead to vendor lock where the DoD becomes dependent on a single service provider because the costs of changing vendors is prohibitive.

In 1994, the Open Systems Joint Task Force (OSJTF) was established to promote the use of open systems architecture from the top-down throughout the DoD. Ever since, there has been an ongoing effort throughout the Department to widely implement open systems architecture. In May 2003, DoD Directive 5000.01 emphasized the use of "modular, open-systems approach … where feasible" (DoD, 2018). Further, in 2004, the OSJTF was identified as the DoD lead for MOSA. Soon after, OSJTF stated that MOSA would be an integral part needed to improve the DoD's joint combat capabilities.

In a January 2015 DoD Directive, program managers were further directed to use an open-systems approach wherever "feasible and cost-effective" (DoD, 2015). Specifically, program managers are to use the Acquisition Strategy to identify where, why, and how MOSA will or will not be used. Once the Acquisition Strategy document is completed, the formal project or procurement can begin. Under the "Business Strategy" section of the Acquisition Strategy, managers are instructed to complete a business case analysis calculation with the help of engineering tradeoff analysis "that outlines the approach for using open systems architecture and acquiring technical data rights." Finally, managers must compare and analyze the results of open architecture to closed architecture to determine which is most cost effective given the military's specific need.

In 2017, the National Defense Authorization Act (NDAA) for FY 2017 amended and formalized the implementation of MOSA (Public Law 114–328; 10 U.S.C. 2446a). Following January 1, 2019, the 2017 NDAA stated that MOSA "shall be designed and implemented, to the maximum extent possible … to enable incremental development and enhance competition, innovation, and interoperability." In January 2019, a memo from the Secretary of the Navy, the Secretary of the Air Force, and the Secretary of the Army emphasized their commitment to MOSA stating "further development of Modular Open Systems Approach standards in areas where we lack them is vital to our success". The most recent NDAA for FY 2020 put responsibility on the Secretaries of the military departments to implement MOSA (Public Law 116–92; 10 U.S.C. § 840). Each branch has responded in slightly different ways to implement MOSA. Broadly, DoD leadership and Congress strongly support MOSA and hope to implement it more completely in the near future.

An open systems approach (OSA), when used in conjunction with a modular architecture, reuse, and/or the harnessing of existing technologies (commercial off-the-shelf [COTS] or proprietary), is commonly associated with cost avoidances arising from more efficient design, increased competition among suppliers, more efficient innovation and technology insertion (faster, cheaper design evolution), and the modularization of qualification. However, the high levels of investment and the increased risk exposure over long system life cycles are often overlooked. Determining if, and to what extent, openness should be pursued remains challenging in the absence of a quantitative model that elucidates the relationship between the degree of system openness and the system's life-cycle cost.

Historically, critical functionality in complex electronic systems was provided by custom-made components and custom proprietary architectures, requiring long development times and high development costs. However, recent technological advancements have allowed for the increased generalizability of both hardware and software (and system architecture); now components can be designed once, and then used in many different applications (Guertin and Miller, 1998). These advancements have increased the viability of using OSA in general, and a Modular Open Systems Approach (MOSA) in particular (Abbott, Levine, and Vasilakos, 2008).

While the DoD supports implementing MOSA whenever possible, there are numerous reasons to be cautious since business and engineering-tradeoffs must be made, possibly changing the incentive structure and reducing the system effectiveness. First, if there are no standards for a new product, then closed system architecture may be best until standards are created (Firesmith, 2015). Second, a poorly designed modularized architecture may be too costly to re-architect and it's better to stick with the old. Third, there is only one qualified vendor to provide the service, making opening the system costly without benefit. These drawbacks challenge the current DoD posture of implementing MOSA throughout the Department, but neither side has successfully provided quantitative analysis to support their position.

Generally, it is taken for granted that the use of OSA and MOSA decreases the total life-cycle cost of a system. Leveraging existing open technology, including COTS components, avoids many costs associated with designing custom systems, and reduces the time required for development or refresh of a system (Logan, 2004). The use of OSA or MOSA helps mitigate the effects of obsolescence, lengthens the system's support life, allows for the incremental insertion of new technologies (OSJTF, 2004) (Boudreau, 2007), and evolving functionality. The use of well-defined standards promotes smooth interfacing both within and between systems, while the proliferation of common component types fosters competition between suppliers. Component design reuse (within and between systems) eliminates redundant components, thus reducing logistical costs.

However, there are costs associated with openness that should be considered. Building a subsystem from open standards and commercially available components often relies on the use of generalized technology with unnecessary, and costly, additional functionality, increasing the cost, complexity, and effective failure rates (Hanratty, Lightsey and Larson, 2002) (Bass et al., 2008). In other cases, it may be necessary to modify COTS components to meet performance requirements (Wright, Humphrey and McCluskey, 1997) (Jenson and Petersen, 1982), thereby adding costs. In addition, the enterprise that manages the system likely has no control over the supply chains for COTS components, which tend to be more volatile than proprietary ones (Lewis et al., 2000). This makes it desirable to refresh open systems designs more frequently (Clark and Clark, 2007) (Abts, 2002), which leads to an increase in the number of fielded configurations, which complicates logistics, resulting in more expensive.

This report seeks to quantitatively analyze MOSA, specifically the relationship between MOSA and life-cycle cost. Throughout numerous DoD documents, some form of life-cycle cost savings is cited a majority of the time as a reason to support MOSA implementation. One study cites preliminary results compiled from 10 years of data on both the Acoustic Rapid COTS Insertion program and its predecessor. These results indicate that lifecycle cost improved by nearly 5:1, but the underlying data is unavailable (Boudreau, 2006). Without quantitative analysis, DoD program managers are left without much guidance on when MOSA is best given life-cycle costs.

Existing Work

Several previous efforts have addressed the measurement of system openness. These include the MOSA Program Assessment and Rating Tool (PART), developed by the Navy's Open Systems Joint Task Force (OSJTF). PART consists of a series of questions, divided into business indicators and technical indicators, that assess the extent to which a certain principle or practice is implemented. PART can be used to measure a system's openness, but it is subjective and qualitative, and the results depend on the optimism with which the survey is completed (OSJTF, 2004).

The Naval Open Architecture Enterprise Team (OAET) developed the Open Architecture Assessment Model (OAAM) to "define, measure, and illustrate the relative levels of openness" (NOAET, 2009). Like PART, OAAM measures the openness of a system using both business and technical characteristics, which are combined to produce an overall openness characterization. While OAAM was relatively simple, it lacked resolution, so OAET created the Open Architecture Assessment Tool (OAAT) to expand and build upon OAAM (NOAET, 2009). OAAT uses a questionnaire that includes PART to assess system openness (NOAET, 2009).

In 2011 and 2012 several parties, including the Air Force Research Laboratory's Multispectral Sensing & Detection Division (RYM) collaborated to develop a set of metrics to evaluate the openness of an architecture. This effort focused on selecting metrics that were broad enough to assess a general case, and quantifiable, so that the measurement would be repeatable. The result of this effort, called the MOSA Metrics Calculator (MOSA, 2012), improves upon the PART questionnaire by ensuring that all metrics are quantifiable.

One common attribute of the PART, OAAT, and the MOSA Metrics Calculator is that they don't account for the cost associated with an open systems approach, and are unable to measure the value of the benefits obtained. They cannot be used to make a business case for the use of openness, i.e., they explicitly assume that increased openness is always beneficial, but is it?

Another approach to measuring openness comes from PMH Systems and the University of Southampton. This work uses an easily quantifiable metric, the fraction of interfaces that use open standards, and a stochastic model to estimate the decrease in cost and development time associated with increasing openness (Henderson, 2009). However, the model implicitly relies on the assumption that increased openness is always beneficial. Additionally, the metric developed cannot resolve different levels of openness and most importantly only addresses the design phase, ignoring significant costs and avoidances that occur later in the system's life cycle.

Research Objectives

Previous efforts have relied on a highly qualitative analysis of a system, with the results often given as an intangible "openness score" used to determine which of multiple system implementations is more open. Such approaches do not provide enough information to make a business case or understand the conditions under which life-cycle cost avoidance can be maximized (or whether there even is cost avoidance). The objective of this report is to quantify the relationship between system openness and life-cycle cost, the key questions the model presented can answer are:

- What are the costs avoided and added due to openness?
- What are the variables that should be considered in assessing the cost impact?
- What level of openness provides the best value to the government/customer?
- How long will a given open system needs to be supported to recover its initial costs?
- How should current policy be modified to ensure an appropriate level of openness is applied?

The discussion of open system pros and cons in the introduction to this report is by no means exhaustive, but meant to point out the complexity that is present in this problem. One possible breakdown of the total cost incurred designing, building, operating, and retiring a system is,[1]

$$C_{Total} = C_{Design} + C_{Production} + C_{O\&S} + C_{Refresh/Redesign} + C_{Enterprise} + C_{Capability} \qquad (1)$$

where Sections 2.2 through 2.7 discuss the modeling of the various contributions to Equation (1).

Section 2 of this report describes a stochastic discrete-event simulation model developed to determine the difference in life-cycle and implementation cost between two versions (scenarios) of the same system (e.g., having different levels of openness). Section 3 provides a case study using the model. In Section 4 we discuss the generalization of the model to consider more general questions about the optimal openness of systems.

---

[1] Some preliminary work done formulating a model based on the Equation (1) appears in (Schramm, 2013).

THIS PAGE INTENTIONALLY LEFT BLANK

Model Development

This section presents a stochastic discrete-event simulation model developed to determine the difference in life-cycle and implementation cost between two versions of the same system (having different levels of openness).

Discrete-Event Simulation Model Development

The model developed follows the life history of a group of items, e.g., a bill of materials (BOM). Life-cycle cost modeling generally involves modeling systems (more specifically system costs) that evolve over time. For complex systems, the time-dependent costs usually involve the operation and support of the system. Depending on the type of system, operation may involve the purchase of fuel, the training of people, the cost of various consumable materials, and maintenance. Maintenance costs that occur over time are combinations of labor, equipment, testing, and spare parts. If the life cycle of a system is relatively short (i.e., less than a couple of years), then direct calculation methods of the life-cycle cost are applicable. However, when the modeled life cycle extends over significant periods of time and the cost of money is non-zero, the calculation of life-cycle cost changes from a multiplication problem into a summation problem and the dates of cost events become important, e.g., the cost of individual maintenance events differ based on when they occur due to the cost of money. Discrete-event simulation is commonly used to model life-cycle costs that are accumulated over time when time spans are long and the cost of money is non-zero.

Discrete-event simulation (DES) is the process of codifying the behavior of a complex system as an ordered sequence of well-defined events. In the context of life-cycle cost modeling, an event represents a particular change in the system's state at a specific point in time, and the change in state generally has cost consequences. Discrete-event simulation utilizes a mathematical/logical model of a physical system that models state changes at precise points in simulated time called events. Discrete means that successive changes are separated by finite amounts of time, and by definition, nothing relevant to the model changes between events. In DES the system “clock” jumps from one event to the next, periods between events are ignored. A timeline is defined as a sequence of events and the times that they occur.

At each event, various properties of the system can be calculated and accumulated. Probability distributions are used to represent the uncertainty in the simulation parameters in discrete-event simulation. This means that we simulate the timeline (and accumulate relevant parameters) with many trials (i.e., through many possible time histories) in order to build a statistical model of what will happen in the life history of the system.

In the model developed for this report (Figure 1) the events of interest are maintenance events, production events (delivery of new systems), retirement events (retirement of fielded systems), logistics events (management of spares, lifetime buys of parts to manage obsolescence) and design redesigns/refreshes. The accumulated properties of interest are cost. The model effectively generates lists of events (date and type of event). The event dates are determined by sampling time-to-failure (TTF) distributions, forecasted obsolescence data distributions, and a predetermined refresh/redesign schedule. The event lists are then pushed through a cost model to accumulate the discounted life-cycle cost.

The description in this section is of the base model. Openness is differentiated by variations in the system components and architecture (and their associated TTF and obsolescence date distributions). The base model does not constitute the entire model. We must also model the relative design and qualification overhead (costs) associated with varying the bill of materials and system architecture (see the next subsection).

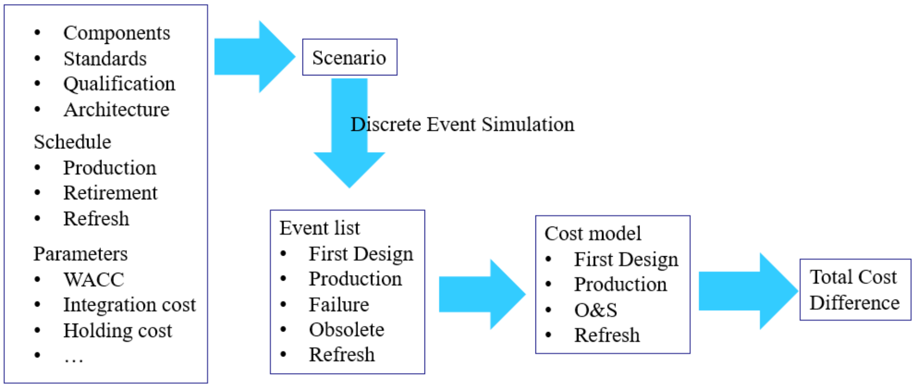

Figure 1 – Cost model used in this analysis.

Design and Qualification Costs (CDesign and CRefresh/Redesign)

C_{Design} are the costs associated with designing a new system that satisfies a set of requirements, and includes the majority of the costs incurred before the final design is selected, including the cost of designing the system, as well as the costs of partial or alternative designs considered, but not implemented.[2] Prototyping and any design overhead costs are also included in the design costs. C_{Design} also includes the Non-Recurring Engineering (NRE) costs, which may include intellectual property costs and testing and qualification costs to demonstrate that the standards, components, subsystems, and complete system meet the required design parameters for performance, reliability, security, etc. These costs may be significantly lower for enterprises that maintain a library of previously used and qualified hardware and software components.

The initial system design represents the effort associated with design and qualification of the whole system. This event only occurs at the beginning of life cycle or when a system redesign that transitions the system architecture occurs. The cost of first design event includes design cost and qualification cost.

Design cost is the cost to design or pick the components and modules for the system. A COTS component has zero design cost while a proprietary component has a design cost determined by user. For a module, the design cost is related to the effort required to integrate all subcomponents in the module, and is proportion to the number of subcomponents. The total design cost is the sum of every component/module design cost.

Qualification cost is the cost to conduct qualification tests for the whole system. Every component and module are assigned required qualification tests. The sum of the qualification cost of all components/modules is the qualification cost.

After the system has been in use for some period of time, it may be desirable (or necessary) to update or refresh the system to ensure it is still manufacturable and supportable. Redesigns that evolve the system functionality and performance may also occur. Refresh costs, $C_{Refresh/Redesign}$, may be similar to those initially incurred in C_{Design},

---

[2] I.e., not used for the current system. The ability to repurpose a previously designed subsystem for a new application is an enterprise-level cost avoidance strategy (included in $C_{Enterprise}$).

except that there is a greater opportunity for design reuse, not only of some components or subsystems.

Production Costs (CProduction)

$C_{Production}$ includes all costs to manufacture and deliver the system, including component procurement, screening and/or burn-in of hardware components, assembly/manufacturing, and any recurring testing costs.

Production events are defined as the action of purchasing components and the resources required to assembly (and functionally test) the system. The production event dates are determined by the production schedule.

Operation and Support Costs (CO&S)

All operation and support costs, including those associated with maintenance, sparing, obsolescence mitigation, and lack of system availability fall under $C_{O\&S}$.

Maintenance (O&S) events occur when there is a component failure. System maintenance includes the labor to conduct the process and the cost to obtain a new component. The model assumes that the system is fixed instantly so there is no downtime associated with a failure. The model also assumes that the maintenance strategy is good-as-new component replacement. If the component is still available in the market, it would be procured as needed.

When the obsolescence of a component occurs, sufficient components are procured and held in inventory in order to support the system until the end of support of the system or the next system refresh (i.e., a lifetime or bridge buy is made) – these components must cover future production and maintenance needs. The cost incurred at the obsolescence event is the procurement cost of the components. Inventory (holding) costs are charged when the parts are taken from the inventory and used for maintenance.

Refresh/Redesign Costs (CRefresh/Redesign)

After the system has been in use for some period of time, it may be desirable (or necessary) to update or refresh the system to ensure it is still manufacturable and supportable. Redesigns that evolve the system functionality and performance may also occur. Refresh costs, $C_{Refresh/Redesign}$, may be similar to those initially incurred in C_{Design}, except that there is a greater opportunity for design reuse, not only of some components or subsystems, but of the overall system architecture as well.

At a system refresh/redesign event, every component/module in the architecture are examined and some components/modules are refreshed due to obsolescence or an update requirement. In the present model, system refresh events are determined before the system life cycle starts.[3] The cost of refresh, including the cost of redesign and requalification, is based on the number of components and modules needed to be refresh, which is the combination of the obsolescence components and the other affected modules/components due to ripple effects. The refresh cost of a component/module is equal to the design and qualification cost described in first design event.

If a component is obsolete when a refresh or redesign is encountered, it is refreshed by replacing it with another component that has the same function (assumed to be not obsolete). In an architecture where components and modules are linked to each other, one obsolete component might affect other connected components and modules, causing them to need to be refreshed as well. When the original obsolete component is replaced by a different component with the same function, it is possible that the new one would have different way to integrate with the surrounding components/modules. Thus, those surrounding components and modules would require redesign and requalification. Further, those refreshed components and modules would affect other connected components and modules, causing a ripple effect. Alternatively, if the component connects to its surrounding components/modules via an open standard, that the component may only require a drop-in replacement that conforms to the open standard. In other words, as long as the standard is not obsolete, the ripple effect would stop when it encounters an open standard.

Enterprise Costs (CEnterprise)

$C_{Enterprise}$ are the costs incurred at the enterprise level, and are shared across all of the different types of systems created by the enterprise, which includes the costs to

---

[3] Models exist that can determine the optimal distribution of refreshes, e.g., (Singh and Sandborn, 2006).

maintain a component library, or any support infrastructure that is shared by more than one system or program.

Capability Costs ($C_{Capability}$)

The definition of capability is “the ability to achieve a desired effect under specified standards and conditions through combinations of means and ways to perform a set of tasks” (DODAF, nd). In the context of this report, the system’s technological capability is its ability to accomplish the “mission” it was created for. For example, a sonar system’s technological capability is determined by performance parameters that may include: range, response time, accuracy, etc. The absolute capability of the sonar system is its effectiveness in detecting objects in the surrounding area, and its relative capability is its effectiveness detecting adversaries or threats early enough to take appropriate action.

If the system’s relative capability is adequately maintained without upgrading the system, then refresh planning becomes a life-cycle cost minimization problem that only seeks to manage technology obsolescence. However, if there is no externally mandated system upgrade strategy, and the system’s relative capability erodes without upgrades, then the erosion in relative capability needs to be considered when planning refreshes.

If capability is modeled as a cost, the cost of system technological capability is not just the cost to implement the capability (which certainly needs to be accounted for), but, more importantly, the costs that result from the capability (or lack of capability). More precisely, the cost is a result of the effectiveness of the system in performing the tasks required of it. For example, failure to upgrade a system’s capability may result in less mission effectiveness or a higher probability of losing the asset do to the actions of an adversary.

Figure 2 shows conceptually how the life-cycle capability cost can be evaluated based on the capability gap between an example system and the state-of-practice of the technologies that comprise the system during the system’s operational life. The shape of the state-of-practice curve in Figure 2 depends on the relevant technologies and their evolution over time. Where state-of-practice represents technology that has matured sufficiently to be practical for incorporation into relevant systems. Δt represents the technology lag time, which is the time it takes to implement a design refresh in the system (design, qualification, and fielding). The larger the Δ*t*, the more likely a system will lag in a capability competition. The area between the state-of-practice curve and the system capability curve is proportional to the total cost of capability during the system's operation life.

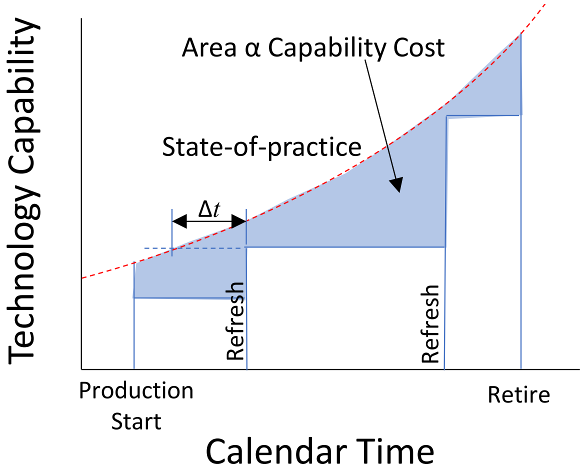

Figure 2 - The relationship between the system's technological capability and capability cost.

As shown in Figure 2, each delivery of a technology refresh to the system resets its capability to a higher level, closing the gap between the system and the state-of-practice capability. Frequent technology refreshes keep the system up-to-date during its life cycle, reducing the probability of losing the technology competition and decreasing the corresponding capability cost. Figure 2 provides a conceptual evaluation of capability cost. Obtaining the quantitative cost, requires constructing a stochastic relationship between capability and cost,

$$C_{Capability} = \sum_{i=1}^{N}\sum_{j=1}^{T} Ip\left(\Delta t_{i,j}\right)C_q \tag{2}$$

where *N* is the number of fielded systems and *T* is the total number of support years for the fleet of systems. *p* is the probability *of* the i^{th} system losing the capability competition in j^{th} year. *I* is the expected number of events (encountering an adversary system) per year per fielded system. Finally, C_q is the effective consequence cost per event.

Comparing Open and Closed Systems – Relative Cost Models

It is often not practical to calculate the absolute value of all the life-cycle costs for a system – this would may be a nearly impossible task. To assess the cost of openness we are only actually interested in the difference in the costs above between two system implementations or architectures. This approach is referred to as a relative cost model (Sandborn, 2019). The advantage of a relative cost model is that all the costs that are a "wash" between the two architectures (i.e., the same) don't have to be modeled because they simply subtract out. The cost difference between two cases is significantly easier to determine than the absolute cost of each of the cases. The model described in this report never produces absolute costs, only the cost differences between two cases.

Figure 3 shows a sample result generated using the model (the data used in this figure will be described in Section 3.1). In Figure 3 and all the figures that follow, year 0 is the year in which the first systems were fielded, i.e., FY99 for the A-RCI. On the left side of Figure 3 are the cumulative discounted life-cycle costs for Scenario 1 and Scenario 2 (which could be two different versions of the A-RCI). In both cases multiple time-history solutions are shown (each is the result of a unique combination of samples from the input probability distributions, so each is one possible time history for the system). The darker solid blue line is the mean of the time histories. The life-cycle costs for the two system implementations are NOT particularly meaningful, because lots of relevant life-cycle costs are left out of the model. The right side of Figure 3 shows the difference between the two scenario costs. The difference between the two cases is meaningful if the costs left out of the model are a "wash", i.e., if they approximately subtract out.

Figure 4 shows the life-cycle cost difference as a function of end-of-support year. The result in Figure 3 corresponds to the end-of-support data point in year 36 in Figure 4. For this example, for an end-of-support year is between 35 to 39, the Scenario 1 is more beneficial (from a cost standpoint) than Scenario 2.

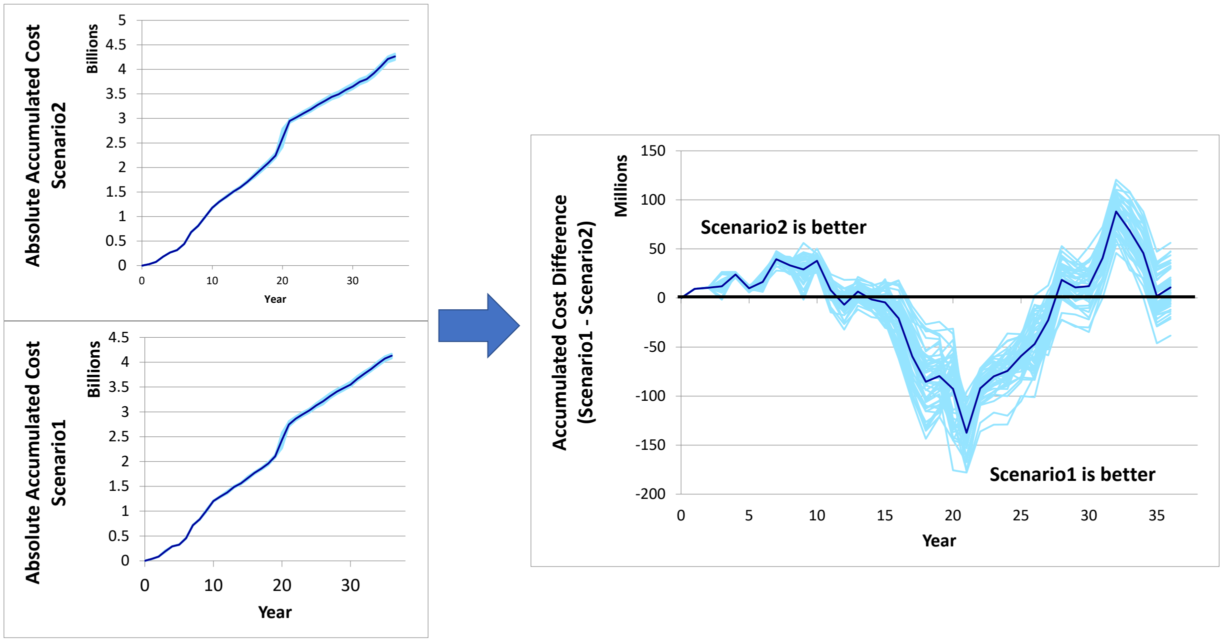

Figure 3 - The relative cost model approach to obtain the accumulated cost difference (end-of-support = 36 years).

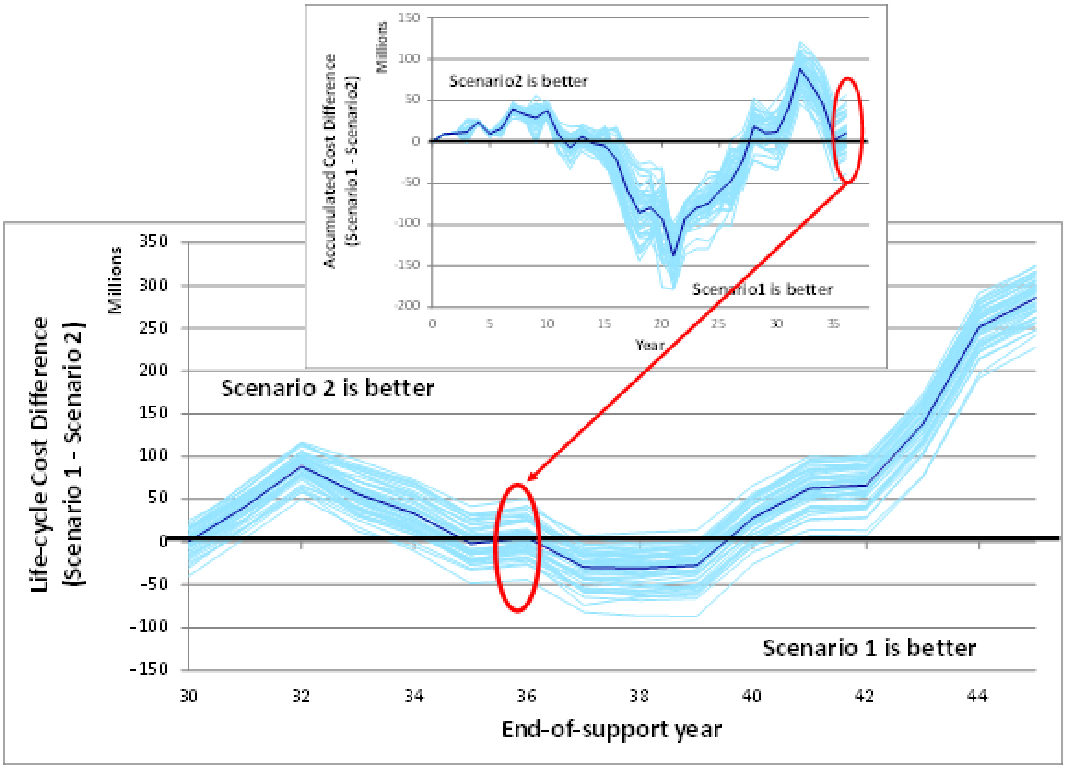

Figure 4 - Life-cycle cost difference as a function of end-of-support year.

THIS PAGE INTENTIONALLY LEFT BLANK

Case Study

In this section we present a case study of the Acoustic Rapid COTS Insertion (A-RCI) Sonar System. The A-RCI program, implemented a COTS-based open architecture for submarine sonar signal processing system. The use of COTS and commercial standards, allowed for the sonar signal processing system to be upgraded, without altering other sonar system components, (Guertin and Miller, 1998). In order to exercise the model described in Section 2, we examined the life-cycle cost difference between two different A-RCI configurations with different degrees of openness.

A-RCI System Data4

Table 1 lists important parameters used in the simulation of the A-RCI. As indicated in the table, some inputs had to be assumed since there was no A-RCI specific information available. Other inputs, as indicated, were determined by calibrating of the model against known A-RCI life-cycle costs (see Section 3.1.1).

---

[4] The data describing the A-RCI in this report represents the author's interpretation of the A-RCI, and may not be representative of the actual system or program. A complete data set describing the A-RCI is not publically available.

Table 1 - Input parameters used for modeling the A-RCI.

Input Parameter		***Value***	***Data Source***
Architecture		Figure 5	Guertin and Miller (1998)
Production and retirement schedules		Figure 6	Schuster (2007)
Architecture R&D cost	Phase I	$48,350,000	From calibration, see Section 3.1.1
	Phase II	$39,015,000	
	Phase III	$55,745,000	
	Phase IV	$56,825,000	
Hardware: COTS/proprietary cards	R&D cost per card type	$2,083,333/$3,125,000	From calibration, see Section 3.1.1
	procurement cost per card	$7,331/$14,545	
	installation cost per card	$733/$1,454	
	reliability	Weibull (1.75,12)	
	procurement life	3 years/6 years	Assumed value
Hardware: infrastructure	R&D cost per infrastructure type	$1,000,000	From calibration, see Section 3.1.1
	procurement cost per infrastructure	$400,000	
	installation cost per infrastructure	$40,000	
	reliability	Weibull (1.75,30)	
	procurement life	20 years	Assumed value
Software	R&D cost	$12,500,000	From calibration, see Section 3.1.1
	procurement cost	$90,909	
	installation cost	$9,090	
	reliability	Weibull (1.75,12)	
	procurement life	3 years	Assumed value
Standards	R&D cost per standard type	$2,000,000	From calibration, see Section 3.1.1
	procurement life	10 years	Assumed value
Maintenance action cost/failure		$38,389	From calibration, see Section 3.1.1
Bridge buy buffer % of demand		50%	Assumed value
Holding cost/component/year		$1,000	Assumed value
WACC		5%/year	Assumed value

Figure 5 shows the assumed A-RCI architecture. A-RCI consists of four primary cabinets, each comprised of one or more VME drawers containing an array of cards. The software connections between the cabinets is indicated as middleware. Open standards are also represented by the linkages shown in Figure 5. The actual number of components (cards) is larger than those shown in Figure 5. The components that are common to all system architectures, whether open or closed, have been omitted – the impact of these components on the life-cycle cost is subtracted out by the relative cost model (as discussed in Section 2.8).

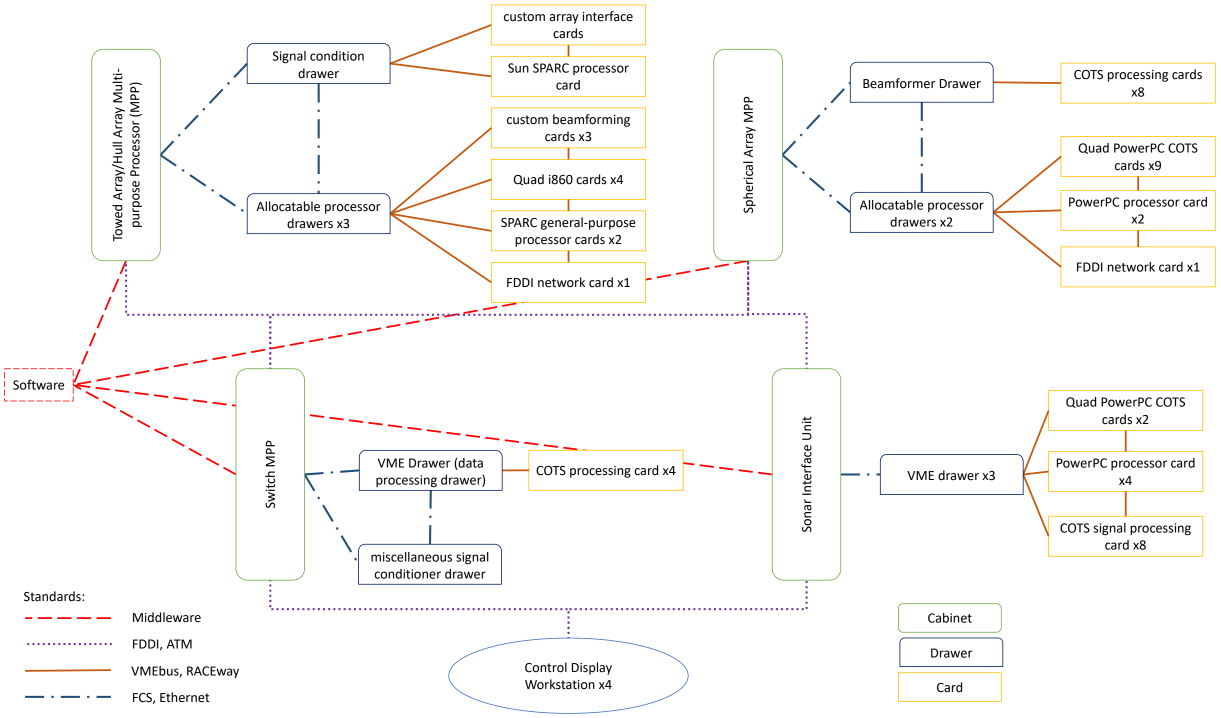

Figure 5 - A-RCI Architecture assumed in the case study (Guertin and Miller, 1998).

A design structure matrix (Eppinger and Browning, 2016) approach was used to capture the interactions between the system's components, which is necessary to cost the design refreshes. During a refresh, when a specific component change is made, the functional and qualification connections to other components are captured by the design matrix and used to determine the entire scope of component changes and re-qualification required.

A-RCI Data Calibration

A-RCI life-cycle cost data was reported in an ASSETT Corporation presentation (ASSETT, 2006). The model presented in this reference, included both a top-down and bottom-up cost estimations based on annual budget and contract data. To determine development and production cost, reliability and procurement life, we reverse-engineered the reported A-RCI life-cycle cost data.

A simple life-cycle cost model was built for calibration. The ASSETT data provides the overall development cost, production cost and O&S cost, we constructed a simple model that calculates these values as a function of known and unknown program and system data. The unknown data in the simple model was then adjusted so that the final costs matched the ASSETT reported values. The results of this calibration process provided in Table 1.[5]

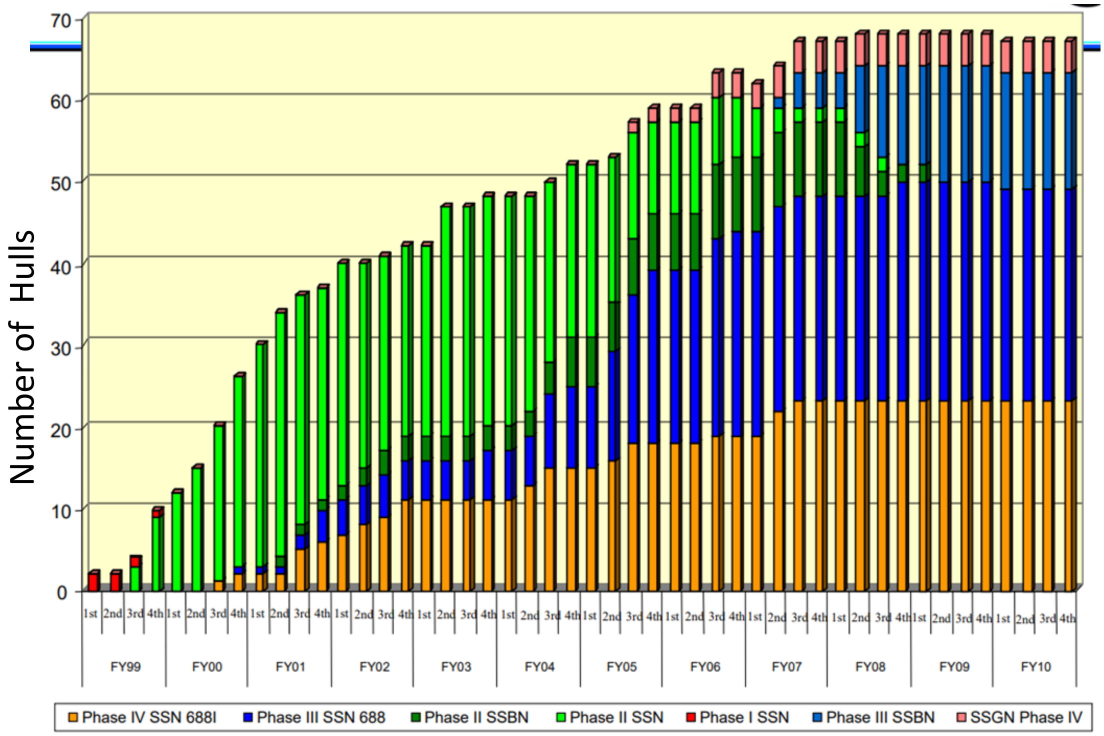

Figure 6 - Installation profile for the A-RCI (Schuster, 2007).

Modeling Results

In this study, the following two system configurations of the A-RCI were compared based on their life-cycle cost:

> "Original A-RCI" = the actual A-RCI architecture, more open.
> "Closed Version" = a less-open version of A-RCI where two of the modules adopted closed standards and proprietary components were used instead of COTS parts.

Both configurations followed the production/retirement schedule in Figure 6 and used the input data described in Table 1.

---

[5] Effectively, we synthesized a data set that uses all the known information and results in the published final program costs. It is unknown if the synthesized data set is unique.

The Impact of Refresh Strategy

The refresh strategy chosen for a system has an impact on both the refresh cost and the capability cost. It is more expensive to refresh frequently, but, more frequent refreshing may reduce the capability cost penalty as shown in Figure 2.

First consider the life-cycle cost of the original (more open) A-RCI system and the closed version assuming the same refresh interval in both cases. Figure 7 shows that the original A-RCI is always more beneficial than the closed version when a 2-year refresh interval are assumed.

The additional results in Figure 7 all assume a 2-year refresh interval of the original A-RCI, but vary the refresh interval of the closed version. The three alternative refresh strategies for the closed version considered were: no refresh, 6-year and 4-year refresh intervals. The closed version of A-RCI is more economical than the original since the no-refresh curve is negative in the end-of-support year range from 30 to 45, i.e., the closed version with no-refreshes has a smaller life-cycle cost than the original version with a 2-year refresh interval.

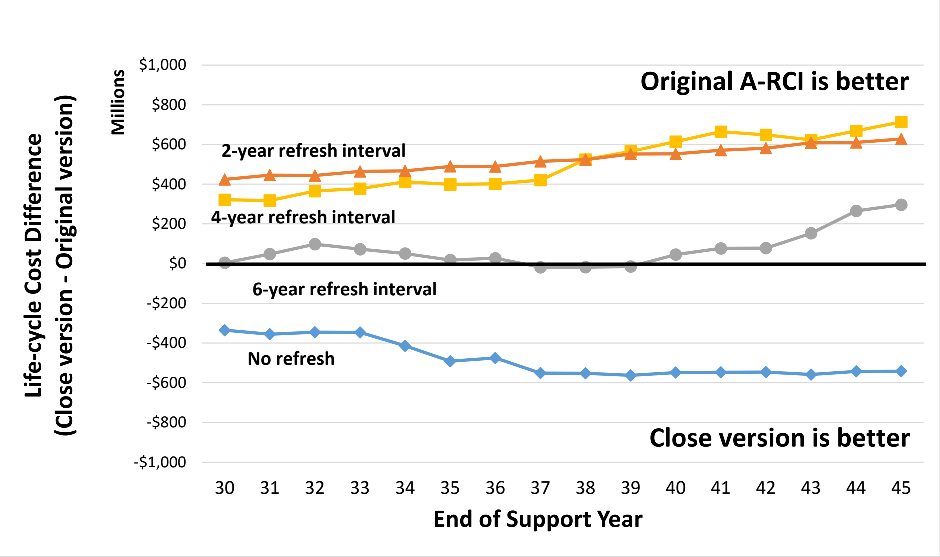

Figure 7 - Life-cycle cost difference given different refresh strategies (C_q = $1M consequence cost assumed).

Figure 7 indicates that more frequent refresh increases the life-cycle cost difference between the two versions. All of the Figure 7 results assume a $1M consequence cost ($C_q$), in this case, for a relatively low consequence cost, the increase in refresh cost (when more refreshes are done) is more significant than the decrease in capability cost penalty obtained from shorter refresh intervals.

Sensitivity Analysis

In this section, we consider the impacts of the risk profile and the fleet size on the relative comparison of the two A-RCI scenarios.

In this case study, the risk is the consequence of losing the capability competition to an adversary coupled with the frequency of encountering an adversary. If the risk is greater (either a higher probability or a higher consequence), given the same system capability, the capability cost penalty will be higher. In this case study, the risk profile was varied by changing the expected consequence from C_q = \$1 million to \$20 million per capability competition loss.

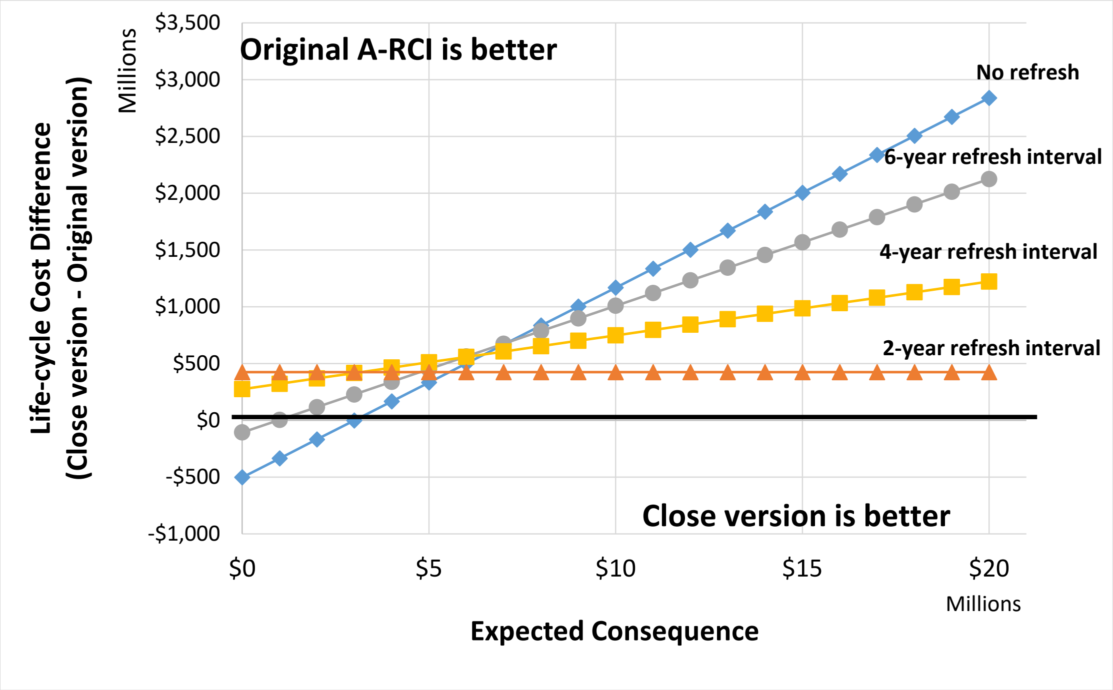

Figure 8 - Cost difference comparison of 30-year support life given different risk profiles and refresh strategies.

In Figure 8 each data point is the mean of one simulation cost difference result. The curves in Figure 8 represent different refresh strategies (refresh intervals) that were implemented by the closed configuration (the original A-RCI always has a 2-year refresh interval). The 2-year refresh interval is a flat line because the two configurations both follow the same refresh interval, i.e., the technology performance would be the same throughout the life cycle in both configurations and there is no resulting sensitivity to the capability cost.

All the results in Figure 8 except the 2-year refresh interval are all increasing functions, indicating that the open (original) configuration becomes more economical as the expected consequence increases. The slopes of the curves decrease as the refresh becomes more frequent. If the expected consequence is over $3 million, it is always more beneficial to adopt an open system approach rather closed approach.

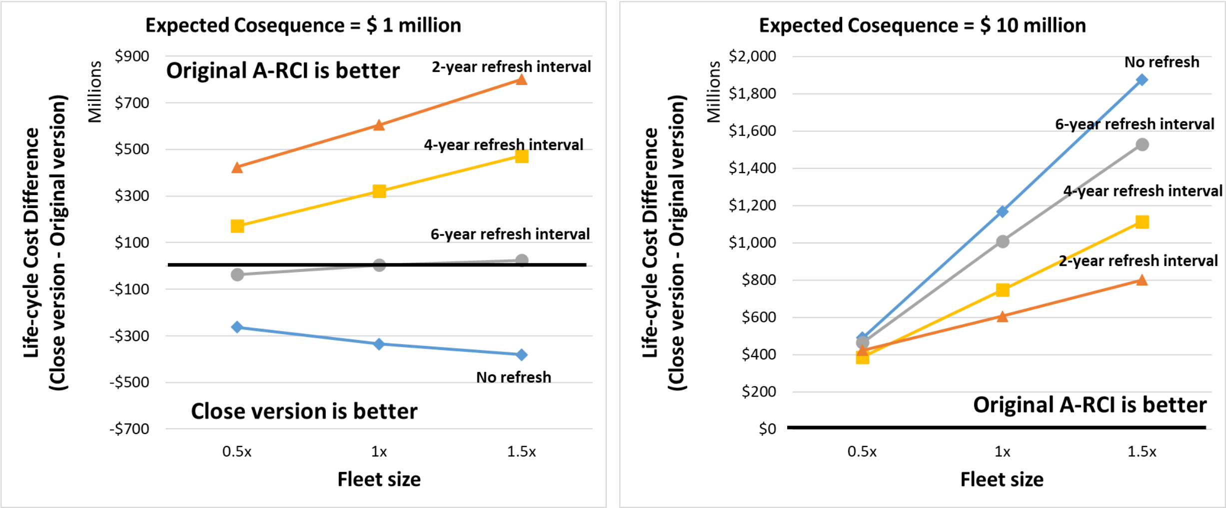

Figure 9 - Cost difference comparison of 30-year support life given different fleet sizes and refresh strategies.

Figure 9 shows sensitivity analyses associated with the number of fielded systems under low and high-risk profiles. In Figure 9, all the systems were supported for 30 years and three fleet sizes were considered: 0.5, 1 and 1.5 times the original A-RCI fleet size. On the left side of Figure 9, capability cost is slightly more dominant in 2-year, 4-year and 6-year refresh intervals. The original A-RCI configuration with more frequent refresh and less capability cost therefore becomes more favorable as the fleet size increases. For no refresh strategy, refresh delivery cost is more dominant, so the closed version with no refresh and less refresh delivery cost becomes more favorable as the fleet size increases. On the right side of Figure 9, the expected consequence is increased to $10 million.

THIS PAGE INTENTIONALLY LEFT BLANK

Discussion and Conclusions

It is often taken for granted that the use of open system architecture (OSA) decreases the total life-cycle cost of a system. However, there is a lack of studies quantifying the cost avoidance and assessing the circumstances under which this assumption is true. This report presents a framework for the quantitative analysis of OSA by modeling life-cycle cost difference associated with system openness, including open architecture, open standard and commercial off-the-shelf (COTS). The cost impact of openness is evaluated by converting these concepts of openness into lifetime events and corresponding costs.

A stochastic discrete-event simulation model was developed to determine the difference in life-cycle cost between two versions of the same system. The model generates events based on system information (BOM, architecture, reliability, etc.) and the consequent cost of each event is then calculated. The sums of total event cost in each of the two versions are differenced to determine which is more beneficial. This model can calculate the cost avoided and added due to openness. The simulation results can also be applied to management strategy optimization.

An A-RCI case study has been used to demonstrate the application of quantitative analysis on cost in relation to system openness. The life-cycle cost difference between the original A-RCI and a less open version of A-RCI was evaluated.

THIS PAGE INTENTIONALLY LEFT BLANK

References

Abbott, J.W., A. Levine and J. Vasilakos. 2008. “Modular/Open Systems to Support Ship Acquisition Strategies,” in Proc. Am. Soc. of Naval Engineers Day, Arlington, VA, 2008.

Abts, C. 2002. “COTS Based Systems (CBS) Functional Density - A Heuristic for Better CBS Design,” in COTS-Based Software Systems, Orlando, FL, Springer, 2002, pp. 1-9.

ASSETT. 2006. Submarine Sonar Cost Analysis Report.

Bass, L., et al. 2008. “Results of SEI Independent Research and Development Project,” Software Eng. Inst., Carnegie Mellon Univ., (Technical Report CMU/SEI-2008-TR-017) July 2008.

Boudreau, M. 2006. Acoustic Rapid COTS Insertion: A Case Study in Spiral Development. Naval Postgraduate School Report. 30 October 2006.

Boudreau, M. 2007. “Acoustic Rapid COTS Insertion: A Case Study in Modular Open Systems Approach for Spiral Development,” in Int. Conf. on Systems of Systems Eng., April, 2007.

CBO. 2019. Long-Term Implications of the 2020 Future Years Defense Program. Congressional Budget Office Report. August 2019.

Clark, B., and B. Clark. 2007. “Added Source of Costs in Maintaining COTS-Intensive Systems,” Cross Talk, The J. of Defense Software Engineering, vol. 20, no. 6, pp. 4-8, June 2007.

DoD. 2015. DoD Instruction 5000.02, Operation of the Defense Acquisition System. January 7, 2015. Available at: https://www.acq.osd.mil/fo/docs/500002p.pdf\

DoD. 2018. DoD Directive 5000.01, The Defense Acquisition System. August 31, 2018. Available at: http://acqnotes.com/wp-content/uploads/2014/09/DoD-Directive-5000.01-Defense-Acquisition-System-31-Aug-2018.pdf

DoDAF. nd. “DM2 Data Groups,” https://dodcio.defense.gov/Library/DoD-Architecture-Framework/dodaf20_capability_mm/#:~:text=A%20capability%2C%20as%20defined%20here,published%20by%20the%20Joint%20Staff

Eppinger, S.D. and Browning, T.R. 2016. *Design structure matrix methods and applications*. Cambridge, MA: The MIT Press.

Firesmith, D. 2015. Open System Architectures: When and Where to be Closed. *Software Engineering Institute Blog*. October 19, 2015. Available at: https://insights.sei.cmu.edu/sei_blog/2015/10/open-system-architecture-when-and-where-to-be-closed.html

GAO. 2014. Review of Private Industry and Department of Defense Open Systems Experiences. GAO-14-617R. June 26, 2014.

Guertin, N.H., and R. W. Miller. 1998. “A-RCI -- The Right Way to Submarine Superiority,” Naval Engineers Journal, vol. 110, no. 2, pp. 21-33, 1998.

Hanratty, J.M., R. H. Lightsey and A. G. Larson. 2002. “Open Systems and the Systems Engineering Process,” Office of the Undersecretary of Defense, Acquisition and Technology; Open Systems Joint Task Force, 2002.

Henderson, P. 2009. “The Case for Open Systems Architecture,” 14 December 2009. [Online]. Available: http://pmh-systems.co.uk/Papers/MOSACaseFor/.

Jensen, F., and N. E. Petersen. 1982. Burn-in: An Engineering Approach to the Design and Analysis of Burn-In Procedures, NewYork: Wiley, 1982.

Lewis, P., P. Hyle, M. Parrington, E. Clark, B. Boehm, C. Abts and R. Manners. 2000. “Lessons Learned in Developing Commercial Off-The-Shelf (COTS) Intensive Software Systems,” Federal Aviation Administration Software Eng. Resource Center Report, 2000.

Logan, G.T. 2004. “The Modular Open Systems Approach (MOSA),” OSJTF presentation to the Exec. Prog. Managers Course: acc.dau.mil/CommunityBrowser.aspx?id=37585, May 2004.

MOSA. 2012. AFRL/RYM Metrics Working Group, MOSA Metrics Calculator, Unpublished, 2012.

National Defense Authorization Act (NDAA) for FY 2017 Public Law 114–328; 10 U.S.C. 2446a

National Defense Authorization Act (NDAA) for FY 2020 Public Law 116–92; 10 U.S.C. § 840

NOAET. 2009. Naval Open Architecture Enterprise Team, Open Architecture Assessment Tool, Version 3.0, User's Guide, 2009.

OSJTF. 2004. Open Systems Joint Task Force, “Program Manager's Guide: A Modular Open Systems Approach (MOSA) to Acquisition,” United States Department of Defense, Available online: http://www.acq.osd.mil/osjtf/pdf/PMG_04.pdf, September 2004.

Sandborn, P. 2017. Cost Analysis of Electronic Systems, 2nd edition, World Scientific, 2017.

Schramm, Z. 2013. A Model for Estimating the Cost Tradeoffs Associated with Open Electronic Systems, MS Thesis, Dept. of Mechanical Engineering, University of Maryland, 2013.

Schuster, J. 2007. “Recent Progress in Submarine Sonar … and Future Challenges in ASW,” MI Department of Mechanical Engineering, Center for Ocean Engineering, Ocean Acoustics Group 7. Available at: http://acoustics.mit.edu/dyerparty/Ira%20Dyer%20talk%20rev%201.pdf

Singh, P., and P. Sandborn. 2006. “Obsolescence Driven Design Refresh Planning for Sustainment-Dominated Systems," *The Engineering Economist*, vol. 51, no. 2, pp. 115-139, April-June 2006.

Wright, M., D. Humphrey and P. McCluskey. 1997. “Uprating Electronic Components for Use Outside Their Temperature Specification Limits,” IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part A, vol. 20, no. 2, pp. 252-256, June 1997.

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

www.acquisitionresearch.net