This product is part of the RAND Corporation monograph series. RAND monographs present major research findings that address the challenges facing the public and private sectors. All RAND monographs undergo rigorous peer review to ensure high standards for research quality and objectivity.

FINDING SERVICES FOR AN OPEN ARCHITECTURE

A Review of Existing Applications and Programs in PEO C4I

ISAAC R. PORCHE III, JAMES DRYDEN, KATHRYN CONNOR,
BRADLEY WILSON, SHAWN MCKAY, KATE GIGLIO, JUAN MONTELIBANO

Prepared for the United States Navy

NATIONAL DEFENSE RESEARCH INSTITUTE

The research described in this report was prepared for the United States Navy. The research was conducted within the RAND National Defense Research Institute, a federally funded research and development center sponsored by the Office of the Secretary of Defense, the Joint Staff, the Unified Combatant Commands, the Navy, the Marine Corps, the defense agencies, and the defense Intelligence Community under Contract W74V8H-06-C-0002.

Library of Congress Cataloging-in-Publication Data is available for this publication.

ISBN: 978-0-8330-5166-0

Published 2011 by the RAND Corporation
1776 Main Street, P.O. Box 2138, Santa Monica, CA 90407-2138
1200 South Hayes Street, Arlington, VA 22202-5050
4570 Fifth Avenue, Suite 600, Pittsburgh, PA 15213-2665
RAND URL: http://www.rand.org
To order RAND documents or to obtain additional information, contact
Distribution Services: Telephone: (310) 451-7002;
Fax: (310) 451-6915; Email: order@rand.org

Preface

This report provides an overview of a study that was intended to lay the groundwork for the U.S. Navy's Program Executive Office, Command, Control, Communications, Computers, and Intelligence (PEO C4I), toward the service-based concept of an open architecture. Building on previous work for PEO C4I that speaks to the flexibility and cost-effectiveness of service-oriented architecture (SOA), the study investigated the potential services that could be used as part of an SOA. Toward this goal, this document presents a summary of recommended services and strategic recommendations for selecting those services for which significant improvements in agility and efficiency are possible. Also included are a number of recommendations selected to fundamentally improve PEO C4I's ability to reconfigure its information technology (IT) system to support change.

This research was sponsored by PEO C4I and was designed with the needs of the sponsor in mind. This document may be of interest to anyone in the Navy involved in the development of services and open architecture standards, as well as the wider U.S. Department of Defense (DoD) community currently considering the benefits of open architectures in terms of aiding the rapid acquisition of IT.

Since the completion of our research, PEO C4I has published a PEO C4I Services Catalog that reflects the description, functionality, and features of C4I services. That document was not finalized at the time of this study.

This research was conducted within the Acquisition and Technology Policy Center of the RAND National Defense Research Institute,

a federally funded research and development center sponsored by the Office of the Secretary of Defense, the Joint Staff, the Unified Combatant Commands, the Navy, the Marine Corps, the defense agencies, and the defense Intelligence Community.

For more information on the RAND Acquisition and Technology Policy Center, see http://www.rand.org/nsrd/about/atp.html or contact the director (contact information is provided on the web page).

Contents

Figures and Table

Figures

Table

Acknowledgments

We are indebted to Chris Miller and Captain Joe Beel (U.S. Navy) for providing guidance on the definition of the problem set to be addressed. Also within PEO C4I, Ed Wunner and Mike Robinson engaged us early with their NITRO concept and thus helped us shape our study tasks. Charlie Suggs provided us with some of the results of his survey of services.

We would also like to thank Chris Gunderson, Naval Post Graduate School, as well as Christopher Priebe and Ralph Carolin of G2 Software Systems, Inc. (supporting PMW 150 and the CFn program). In addition, we interviewed Bob Poor, Jeff Jones, and Brad Garey, DCGS-N; Terry Simpson, Space and Naval Warfare Systems Command (SPAWAR) Systems Center; Ron Gray, Tom Klooster, and Otis Glover, NTCSS; Eric Helgeson, DIO-S; Victor Leung, MEDAL; Jeremiah Hayden and Brett Hayes, SPAWAR 1.6; Delores Washburn, Femi Adeyemo, Mary Nies, Andy Farrar, Joaquin J. Martinez, Stephanie Meyers, and Lysa Olsen, PMW 160; Sean Moone and Rob Peabody, GCCS-M; Stacie Sherwood, Application Integration; and John Isaacson regarding SPIDER data. John Hong and Mark H. Davis engaged us in numerous discussions and provided us valuable reports. At RAND, John Birkler provided input and guidance on the project effort. Phil Antón and Mark Arena provided reviews of this report and initial project plans. Kate Giglio served as the project communications analyst. Finally, Michelle McMullen helped format and edit this document and Sarah Hauer provided some additional administrative support.

We are most grateful to our reviewers, Jeff Rothenberg and Steve Sudkamp, who provided expert advice, guidance, and input.

Abbreviations

C4I	command, control, communications, computers, and intelligence
CANES	Consolidated Afloat Networks and Enterprise Services
CFn	Composeable FORCEnet
CNO	Chief of Naval Operations
COTS	commercial off-the-shelf
DCGS-N	Distributed Common Ground/Maritime System–Navy
DIO-S	Distributed Information Operations–Services
DoD	U.S. Department of Defense
FMEA	Failure Mode Effects Analysis
GCCS-M	Global Command and Control System–Maritime
HM&E	hull, mechanical, and electrical
IT	information technology
MEDAL	Mine Warfare and Environmental Decision Aids Library
NTCSS	Navy Tactical Command Support System

NTRM	Navy Technical Reference Model
ORB	Object Request Broker
OTH-GOLD	Over-the-Horizon GOLD
PEO	Program Executive Office
PMW	Program Manager, Warfare
REST	Representational State Transfer
ROI	return on investment
SCN	Ship Construction Navy
SLA	service-level agreement
SOA	service-oriented architecture
SOAP	Simple Object Access Protocol
UAV	unmanned aerial vehicle
UDDI	Universal Description, Discovery, and Integration
WSDL	Web Services Description Language
XML	Extensible Markup Language
XSD	XML Schema Document

Finding Services for an Open Architecture: Existing Applications and Programs in PEO C4I

For decades, the Navy's command, control, communications, computers, and intelligence (C4I) systems have provided commanders and sailors alike with information vital to situational decisionmaking and command. Maintenance for and upgrades to the numerous sensors, receivers, computers, and networks that make up these systems, however, have proved a distinct challenge to program offices. The aging of shipboard networks and applications continues to grow at a rapid rate as a result of commercial technological advances. Program Executive Office (PEO) C4I—along with the entire Navy—is aware of the need to better integrate computer and application updates and improvements with respect to timeliness and cost-effectiveness. The cost of maintaining legacy and often stovepiped networks continues to grow at a rate that is negatively affecting the procurement budgets for new systems. Staying on the current path will cost the PEO C4I $2.6 billion (in FY2007 dollars) over 15 years just to maintain current capacity—a figure that suggests that PEO C4I needs to better leverage its investment in developing technologies. A service-based, open architecture (e.g., a service-oriented architecture, or SOA) has been proposed as a means to help offset costs and maintain upgradability.

In memoranda defining his "Agenda for Change," then Deputy Chief of Naval Operations for Communication Networks (CNO-N6)[1] VADM Harry Harrison, Jr., articulated a vision of better information technology (IT) governance that can bring about standardized Ship

[1] Since the publication of this memo, N6 and N2 have been merged.

Construction Navy (SCN) C4I environments and standardized procedures (Department of the Navy, 2008).

This is to occur through improved processes to define ship C4I requirements and standardized C4I acquisition plans. N6 noted that a challenge to achieving the vision lies with platform programs[2] that independently select unique C4I solutions, resulting in the following outcomes:

- increased total ownership costs (including additional costs required for unique integration during technology refresh)
- inability to leverage economy-of-scale benefits
- lack of interoperability with existing programs of record and resulting early obsolescence
- higher upfront costs for new proprietary systems
- configuration management challenges among platforms
- lack of a program of record to support unique configurations and thus no life-cycle support plan for C4I.

The goal of this project was to assess applications and services within the purview of PEO C4I that could be utilized as part of an SOA and to determine the extent to which this helps ameliorate the negative impacts highlighted by the CNO-N6 of today's acquisitions and acquisition strategies.

Study Objective and Purpose

Our objective in this study was to identify current and planned systems (e.g., applications) within the PEO C4I portfolio that can contribute to or benefit from utilization of an SOA. We also endeavored to provide

[2] Others argue that this problem runs deeper than platform managers and extends to resource sponsors and different types commanders (who determine requirements). Also, another problem is that the Navy does not "think or act as an enterprise," though there are some sub-enterprise examples within the Navy (e.g., nuclear propulsion, submarines, Navy Air) (Sudkamp, 2010).

analysis that quantifies additional costs and performance benefits from expanding the set of candidate applications that can utilize it.

The study focused on the PEO C4I portfolio, though one program in PMS 495 was examined. Our objectives did not include the larger question of SOA realization for the Navy, but we do acknowledge the fact that SOA requires incremental steps for realization. Our study focused on the incremental step of identifying and categorizing services within the PEO C4I portfolio. We were also asked to identify SOA barriers during our service categorization efforts.

Finally, we examined the intersection between life-cycle costs and capability in the Navy and in the private sector. Factors that affect cost include hardware-centricity, whether there is an open architecture or a specialized system with few vendors, and the level of standardization of infrastructure. The general trend is that life-cycle costs will continue to decline in the private sector as capability increases over time, but the opposite will be true of Navy C4I: Its costs will increase as capability increases.

Design of This Study

This document provides an overview of recommended services and strategic recommendations that will assist PEO C4I in moving away from its current, stovepiped network system, which, at any given time, meets the needs of only a limited number of users. To identify those services that will help create greater interoperability between systems and users in a cost-effective manner, we addressed the following questions:

1. How can existing applications within the PEO C4I portfolio be categorized as services?
2. What are the services or near-services available for incorporation or maturation into a service-based, open architecture?
3. Which of these services provide the most value to PEO C4I?

4. How much does it cost to develop a service?
5. How can the service-based, open architecture be made to work?

General Approach

To evaluate the suitability of key programs to contribute and/or accommodate services as part of an open architecture, we reviewed existing reports and requirements documents from the Navy, including the PEO C4I Masterplan documentation, application costs reports, and technical information briefings for numerous programs of record. Such reports allowed us to understand what applications were most important and what common areas of capability existed. We also conducted interviews with subject-matter experts from the various program offices. In these interviews, we were most interested in learning what services had been developed, what barriers existed for service development and adoption, and what future plans existed for using and creating services. The review of the literature and the interviews together allowed us to address the aforementioned questions.

The data gathered from the technical reports and interviews allowed us to apply a number of analytic techniques to measure cost and analyze PEO C4I SOA potential from our unique service categorization, described below. We developed a cost model, and we present statistics on the fit of the model in the full report provided to the sponsor.

Specific Steps

The specifics of our approach to identify existing services of value are outlined as follows:

Step 1. **Identify a useful taxonomy** that enumerates the range of core and/or application services deemed valuable by PEO C4I and the larger Navy community. This involved use of the existing Navy Technical Reference Model (NTRM) as well as the development of new categorizations based on maturity, utility, and complexity of the software components involved.

Step 2. **Characterize existing PEO C4I systems** in terms of the services they could provide. This is based on the taxonomy

in step 1, which involves populating an NTRM-based matrix.

Step 3. **Identify a more limited set of candidate systems** to be considered for our analysis based partly on the characterizations in step 2 and other reports and surveys, in addition to recommendations and input from PEO C4I.

Step 4. **Develop and utilize system attributes and metrics** to score the candidate applications' "SOA-ability" and utility. Based on interviews conducted in person or via telephone, we developed a general set of questions and filled out a detailed checklist/matrix.

Step 5. **Gather additional detailed data** as needed (via interviews and other means) using the instruments developed in steps 1 and 4 for the candidate applications selected for focus in step 3.

Step 6. **Evaluate** the applications' cost and benefit (including technical feasibility of making an application a service) and make recommendations. This is done by addressing the following question for each application identified:
- To what extent can the service be shared/reused?
- What is the cost to develop/mature such a service?
- What is the cost to support such a service?

Cost Metrics Considered

Industry and government are still developing metrics to accurately measure the benefits and performance of the SOA paradigm. No performance and cost benefits for SOA have been accepted industry-wide (at least not yet). Nonetheless, there are some general notions that should be adhered to when considering metrics. One is that properly designed metrics should align outcomes with business objectives.[3]

The total cost and cost-effectiveness of transitioning to a more open, networked approach has not been studied comprehensively, but

[3] Theoretically, with appropriate metrics IT managers can evaluate whether a solution will provide better outcomes for the business than existing networks and consider tradeoffs among alternatives.

related cost efforts exist. One such program, Consolidated Afloat Networks and Enterprise Services (CANES), anticipates significant life-cycle cost savings over status quo because of the reduced number of networks and increased standardization across ships, which should reduce labor costs. Recent CANES briefings focus on current costs versus the CANES functionality and anticipate savings. Full life-cycle costs of the program, including operating and support costs, require additional analysis.[4] While the services identified by this study are not part of CANES, the team used the CANES processes as a case study on how a new set of applications would be transitioned to a service. The RAND team worked with PEO C4I in identifying sources of data that estimated costs associated with selecting and adapting applications for SOA services. Figure 1 illustrates the starting point of the cost analysis and the effect of the effort expended.

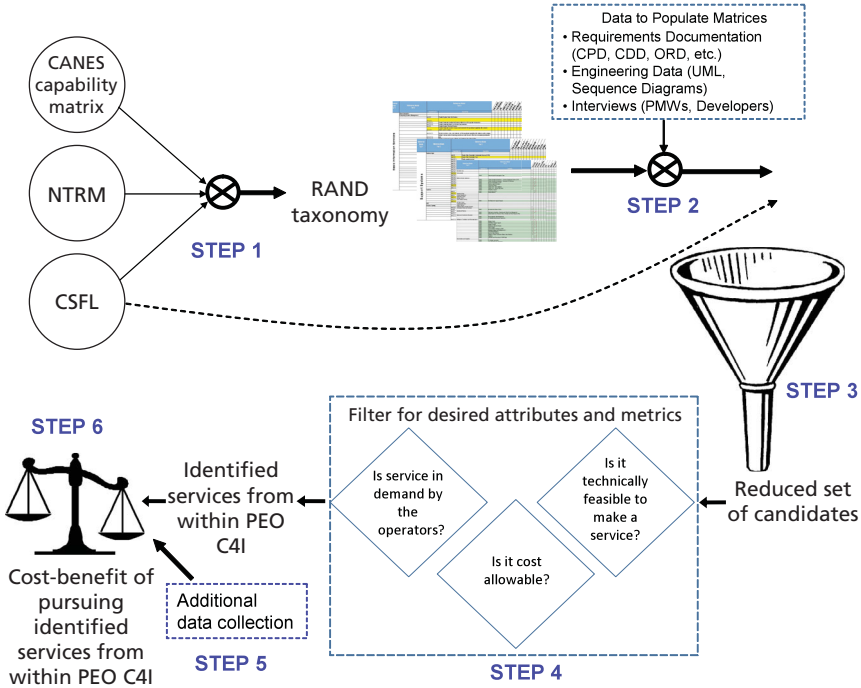
What Is a Service?

The U.S. Department of Defense (DoD) Architecture Framework defines a service as "a distinct part of the functionality that is provided by a system on one side of an interface to a system on the other side of an interface" (U.S. Department of Defense, 2007, p. B-5; derived from IEEE 1003.0). IEEE 1003.0 further states,

> A service is also a callable routine that is made available over a network. A service exposes an interface contract, which defines the behavior of the service, and the messages it accepts and returns. It is a unit of work done by a service provider to achieve desired end results for a service consumer.

[4] Recent research by the RAND Corporation on C4I upgrades (Schank et al., 2009) indicates that early adopters of CANES have experienced higher-than-average labor costs (within the ship class) for typical C4I upgrades, but Shank et al. do not have data for the actual CANES roll-out or other SOA-related initiatives.

Figure 1
Steps in Approach



NOTE: CSFL = common system function list.
RAND *MG1071-1*

In this study we assert that an "ideal service" is a software component that executes a task and has the following properties:

- is a run-time entity that is always "alive" on some server
- is self-contained/loosely coupled
- has predetermined functionality
- can be exposed through a standard interface
- is discoverable and self-describing
- is part of a service-based, open architecture.

The motivation for this research is the assumption that programs within PEO C4I may already provide software components with some or all of these service-like properties.

What Is a Service-Oriented Architecture?

A service-oriented architecture (SOA) is an architectural style. It is simplest to say that it is an architecture with service providers and service users/consumers. There are many other, more technical and more verbose definitions. For the purpose of this section, we use the following definition:[5]

> SOA: A set of principles that together define a network architecture at runtime that is loosely coupled and comprising discoverable service providers and service consumers that interact according to a negotiated standard or interface.

For PEO C4I's interests, an SOA is at least two things: (1) an approach to integrating current (legacy) and planned (afloat) applications and (2) a means to facilitate the increased utility of SOA-compliant commercial off-the-shelf (COTS) applications (on networks that might not already be amenable to COTS utilization). The first utility is vital because, although the general idea of an SOA is relatively new, the individual service functions that comprise it may already exist in the form of current and near-term applications.

How Can Existing Applications Be Categorized into Services?

A consistent and sound open architecture requires that services first be consistently defined in a way that gives precedence to the governing user (in this case, PEO C4I). There are a number of dimensions to consider when describing services, and each of those dimensions has

---

[5] The definition provided in the CANES SOA RFI (rev. 2) is tracked and as follows:

> A software architecture that enables business agility through the use of loosely coupled services. Services are implementation-independent reusable business functions that are discovered as self-describing interfaces, and invoked using open standard protocols across networks. Services can in turn be combined and orchestrated to produce composite services and business processes, in accordance with pre-defined policies, security, and SLAs.

many degrees. Three of them we considered at length: maturity, utility, and complexity. But there is also robustness and vetting (via "beta-testing"), mission criticality, security, and others.

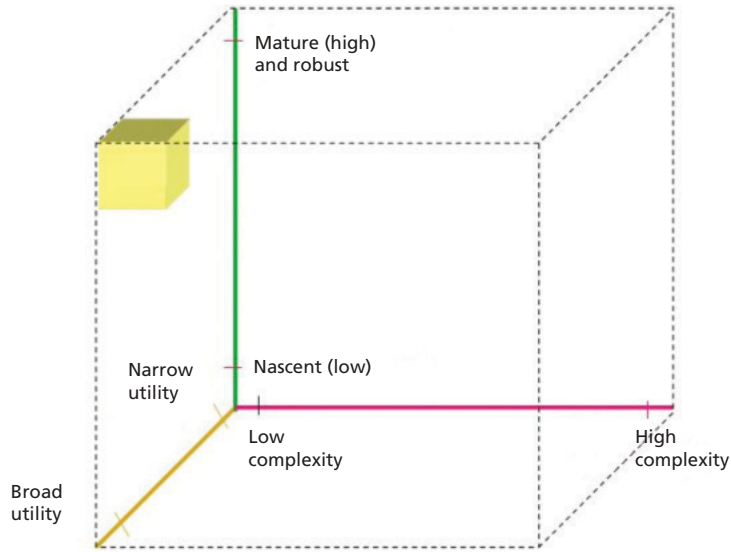
Figure 2 illustrates how an ideal service aligns along the three dimensions we discuss, i.e., a broadly applicable and sufficiently robust service matured to a high level with a low amount of complexity.

- **Utility**—from small building blocks to mission applications
- **Maturity, or "CANES-ability"**—from "conceptual" to "Consolidated Afloat Networks and Enterprise Services (CANES)-ready"
- **Complexity**—from simply developed components to extensively planned software development.

Maturity ("CANES-Ability")

There are many degrees to each service dimension, and where the lines are drawn between each degree is the basis for the categorization. For

Figure 2
A Perfect Service



RAND *MG1071-2*

maturity, also known as "CANES-ability,"[6] our approach was based on a look at the technical standards and techniques being employed by the developers of the service. Maturity is not meant to measure a service's intrinsic value, just its ability to fit well in a well-defined, service-based, open architecture. We discuss the value of a service later in this report.

Table 1 describes the levels of service maturity we used. A service at the conceptual level, Level 0, is hardly a service by all practical standards, but it represents an existing or discrete function of an application or system. Nascent services, at Level 1, build on Level 0 but are also modular and at least somewhat loosely coupled. The use

Table 1
Maturity Model/CANES-Ability Rating

Level	Description
L0: Conceptual	an *existing distinct/discrete function* within an application or system
L1: Nascent	modular or *somewhat loosely coupled*
L2: Functional	a *common data format* (e.g., XML Over-the-Horizon [OTH]-GOLD) that uses a common definition language (e.g., XML Schema Document [XSD])
L3: Mature	a common format for defining interfaces (e.g., Web Services Description Language [WSDL], XML), with a published interface (*self-describing*), a common messaging format (e.g., Simple Object Access Protocol [SOAP], Representational State Transfer [REST]); architecture *supports* interaction with applications servers (e.g., JBoss, WebSphere) or has an internal suite of services
L4: Mature Enterprise	a service that can be *invoked using common mediation*, i.e., designed to be fully compliant with a set of standards of an *enterprise service bus (ESB)* concept, a common registry mechanism (e.g., Universal Description, Discovery, and Integration [UDDI], Object Request Broker [ORB]), or some function that enables exposure of service metadata
L5: Ideal CANES	*compliant with all standards associated with the PEO C4I Stack* or ONR RI Stack of protocols; a service that can be invoked over the ESB provided by the Afloat Core Services (ACS)

[6] We note for clarification that "CANES" or "CANES-ability" is not a standard or set of standards; this "CANES-ability" maturity ranking is used to classify a service's ability to fit well in a well-defined, service-based architecture that may or may not reside in the CANES environment.

of the term "somewhat" in the definition highlights the importance that, during the course of characterizing the services, any placement into a particular level was open to interpretation. At the functional level, Level 2, services are discrete, modular, and also use a common data format, such as Extensible Markup Language (XML), for defining data. Level 3, mature services, is where the concepts of SOA and Web services begin to take shape, and what we categorize as a mature service is what most would consider a modern software service. These include a common format for defining interfaces, a published interface, a common messaging format, and an architecture that is compatible with today's application servers. We did not distinguish between specific technologies used for either Representational State Transfer (REST) or Web services (WS-*) implementations, though a vast majority of programs examined were adhering to WS-based standards. At Level 4, mature enterprise, the service can be invoked using common mediation—that is, it is designed to be compliant with a set of standards of an enterprise service bus, a common registry mechanism, or some function that enables exposure of surface metadata. Finally, at Level 5, the ideal CANES services would be compliant with whatever standards, data model, and software architecture have been prescribed by PEO or inherited from the Navy or DoD.[7]

Utility and Breadth

A second dimension of services is their utility. Some of the legacy software systems within PEO C4I have a broad utility, e.g., broadly applicable utilities that are enabling services such as data discovery, data retrieval, visualization, etc. In contrast, some software is more narrowly focused, such as software designed to meet a specific warfighter need, and is often designed to be user-facing. For example, an advanced common operating picture tool for a specific user is a narrowly focused function.

Categorizing these services by utility helps put them in the context of more generic capabilities, such as those described in the NTRM:

---

[7] Level 5 maturity is more likely to be interoperable, though it does not guarantee all forms of interoperability (such as "semantic interoperability").

application services, common services, and cross-cutting services. The terminology we used is closely related; we categorize services into three types of utility:

- **Mission Services**
  - focused to deliver specific warfighter need
  - application level functionality; may, but not required to, interface with users; possibly built on other common and building-block services (sometimes called composite)
- **Common/Core Services**
  - enabling services that help meet warfighter needs, e.g., visualization, data retrieval, data discovery, etc.
  - machine-to-machine services, more complex than building-block services
- **Building-Block Services**
  - meets generic needs
  - simplest form of a service; provides data access; broad utility for common standards.

An example of a mission service would be a Web-accessible user interface that provides the warfighter situational awareness from signal intelligence sources. This mission service could depend on a number of common/core and building-block services. Several common/core services, such as an IO portal and a query service, might be required to run this mission service. Also, several building-block services, including data mediation, security, and messaging, might be consumed by this mission service.

Figure 3 shows a framework for tabulating services along the two dimensions of maturity and utility.

Complexity

Coding complexity is another important dimension of a service. The more complex a service gets, the more difficult it can be to realize in a service-oriented environment. We considered three levels of complexity for software services: low, medium, and high. Highly complex services are those that are most dependent on clear architecture docu-

Figure 3
Framework for Tabulating Services with the Lexicon Introduced (values are notional)

Utility / Maturity	Mission-level utility	Small, building-block utility
Ready for CANES	1	10
Conceptual-only maturity	20	100

RAND *MG1071-3*

mentation, data reference models, and mature software engineering processes. We measured the complexity of services according to the following descriptions:

- **Low**—software-coding effort is straightforward; any competent junior coder could sit down and write the code as fast as he or she could type, with minimal debugging effort.
- **Medium**—requires more significant planning and coordination.
- **High**—requires experienced developers who utilize careful planning and extensive coordination with others to develop coding strategies.

Robustness, Mission Criticality, and Security

While we did not capture this information in this study, robustness and vetting (achieved perhaps via extensive "beta-testing") remains an important element to measure in assessing services for SOA implementation. Further study of services can reveal if a service has:

- **High Robustness—**tested for scalability in laboratory and production environment and has ability to remain operable despite abnormalities

- **Low Robustness—**untested, potentially fragile or buggy, and unclear scalability.

The Navy also has an added dimension of mission criticality. This means that no matter how costly or inefficient a service may be, if it uniquely satisfies a mission need, then it may be necessary to develop that service.

Another element not considered, but still highly important, is security or information assurance.

Service Suitability

Another SOA concept we address with the lexicon is service suitability. Service suitability is the capability of a service to perform a set of coherent, isolatable actions. Suitability is concerned with how broadly applicable the service is and how isolated its functionalities are. For example, if a service depends on many external functionalities and/or data, it may not be a good candidate as a service.

NTRM Tiers. The NTRM uses a tiered approach to classify software functionality that supports afloat, airborne, and ashore Navy platforms. Each progressive tier provides further breakdown in the functionality. For example, geospatial visualization is a Tier 3 functionality in the NTRM. Within this Tier 3 functionality are Tier 4 functionalities we developed that identify more-specific geospatial functionalities. An example is a terrain analysis function.

Using NTRM Tiers to Gauge Service Suitability. Using the NTRM, we assessed how applicable a service is by matching each service's functionalities to the appropriate NTRM Tier 4 functionality. This NTRM analysis measures the applicability of a service by its reuse potential among the Navy programs we examined. We do not measure how isolated the services functions are, but it is argued that this NTRM analysis provides a structure to make this assessment.

Modeling

Cost analysis in the software arena continues to be challenging because of difficulties in expressing the scope of a project, limited data availability, wide ranges of programmer capability, and other factors. Analysis of SOA service development proved to be no exception.

Nevertheless, we were able to build some cost-projection models using this limited data to provide some preliminary perspectives on the range of costs involved and to develop and demonstrate how these models work. These models rely on the limited data available from case studies combined with a select number of variables that could be assessed on a wide range of services.

We were able to estimate the life-cycle cost of a service (including development costs) using a regression analysis of the data received and categorizations by service type, CANES-ability rating, and complexity. The model results show that total life-cycle costs, as well as just development costs, increase with maturity and decrease with complexity. The challenge for the Navy with legacy systems is that they are expensive to continue operating over time. Developing new systems is also expensive but may result in easier-to-operate systems with low operations costs. One SOA cost element that we did not consider in our analysis is service refactoring. Since SOA realization requires incremental steps, legacy services will require ad hoc interfaces to operate in the emerging SOA environment.

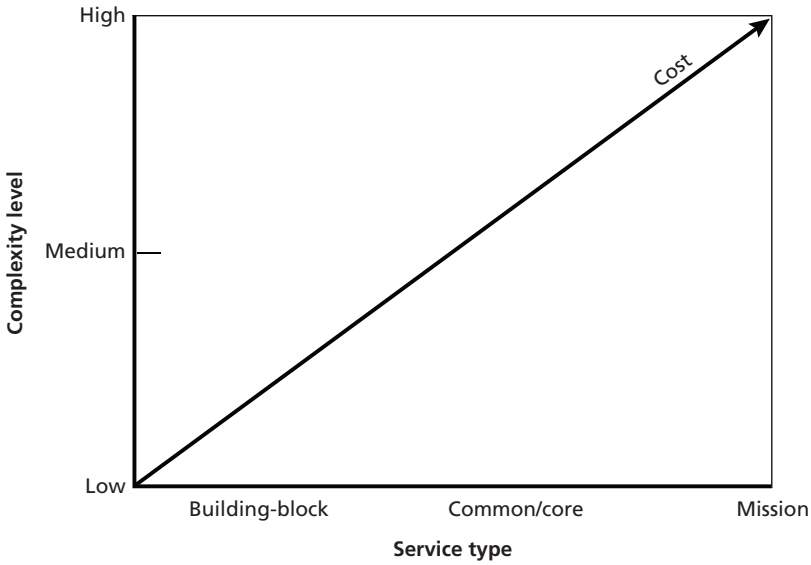
Figure 4 shows the trends in the regression coefficient for the complexity and service type components of the estimated service life-cycle costs.

Maturing a low-complexity or medium-complexity common/core service would pay off if it was expected that at least two more uses of this service would occur. It is economically reasonable to mature these services if reuse is anticipated and possible.[8]

One of the initial activities required to achieve an SOA is to categorize the current and planned services in the organization. From this

[8] A service could be uniquely catering to a specific environment and thus not reusable outside of its intended environment.

Figure 4
Trends in the Regression Coefficient for the Complexity and Service Type Components of Estimated Service Life-Cycle Costs



RAND *MG1071-4*

categorization, areas where there is potential reuse of services among Navy programs can be identified for further analysis. This is only one criterion that can be used to measure potential SOA benefits in an organization. The NTRM overlap analysis identifies the generality or customizability of a service for potential reuses among the Navy programs examined. Reuse potential among programs in this analysis is concerned with separate programs with current software or planned software development that have similar functionality. The NTRM analysis identified more than ten opportunities for reuse between two or more programs for mission services and high-complexity common/core services. From this analysis of the cost of maturing a service and potential overlap, it appears that the greatest cost benefit occurs for medium- and high-complexity mission services and high-complexity common/core services.

Other Implementation Issues

There are several implementation issues with respect to the SOA concept that deserve mention:

- Legacy issues: In the transition to an SOA environment, there is a need to maintain legacy interfaces in parallel with the SOA development.
- Bandwidth and security: The tactical boundary poses bandwidth and security issues. Any implementation will have to be mindful of possible limitations.
- Latency: Performance issues may arise for real-time processes that are transferred to a remote location for processing, when that is the case (Rothenberg, 2010).

Making a Service-Based, Open Architecture Work

A logical but challenging objective of open architectures is to cut lifecycle costs and enhance interoperability. Open architectures (including the specific idea of an SOA) enable this by standardizing interfaces and enabling a common architectural framework. Among other benefits, new information exchanges are expected to develop. However, critics of the idea of an SOA have identified potential limitations to the SOA concept. These viewpoints come from the diverse perspectives and personal agendas of their creators. We examine some of the issues, including network bandwidth limitations, security implications, the source of the standards used, interoperability, semantics, and organizational issues.

Challenges

Entrenched Legacy Systems. The Navy is a mix of legacy and modern systems, just as are many commercial organizations. A recent Center for Naval Analyses report talks about the evolution of software systems and the need for modernization:

> Not only are these systems unable to interoperate with each other (which is usually the case, since they were never built to interoperate with other systems; rather, they were built to perform some operation), they often contain redundant and/or conflicting information (compared to the organization's other systems). As such, it is desirable for an organization to merge its IT systems (so that data from different divisions can be available company wide) and eliminate duplicative or conflicting data and processes. (Tsui and Shea, 2009)

Legacy Organizational Limitations. PEO C4I has separate organizations representing each functional area. The areas are generally understood and make sense in an organizational chart, align similar mission competencies, and have evolved working relationships to share information and objectives. Programs are funded based on this design. Unfortunately, this setup is not the best for a net-centric operating environment, and the PEO recognizes this:

> . . . many Programs are focused on building "systems" vice "services" that provide capabilities that contribute to the evolution of the Global Information Grid (GIG). In a "system-centric" acquisition environment, there are clear lines of authority and funding that foster a myopic view of the enterprise and lead to building stovepipes that do not interoperate. (PEO C4I Master Plan 3.0, 2009, preface)

Need for Standards. An underlying principle of SOA is that services are loosely coupled. This loose coupling is realized by the services adhering to agreed-upon standards or a standards "stack." Without adherence to these standards, each service interaction will require a unique coupling, which prohibits interoperability and reuse of software services.

The issue of standards often dominates SOA discussions. While important, simply declaring the necessity of standards (or the intention to use standards) will not by itself enable an SOA (Tsui and Shea, 2009). There are a few reasons for this: The same standard can be invoked differently; not all standards are interoperable (e.g., "Just

because you're standards-based and I'm standards-based doesn't mean we're interoperable"); and many open standards have multiple (and incompatible) ways that they can be implemented, and such flexibility in implementation can cause interoperability problems in COTS (Tsui and Shea, 2009).

Generally speaking, the government does not try to pursue the setting of standards; there is a preference for letting these come from industry. However, in industry, standards are constantly being revised, e.g., relevant commercial standards such as SOAP, WSDL, and UDDI are under revision (National Research Council, 2006, p. 135; Tsui and Shea, 2009).

Solutions

Routinely Perform Domain Analysis. It is well accepted that considerable upfront costs and effort are required to realize an SOA, thus requiring the organization to function properly to gain this capability. A critical initial step for SOA realization is for an organization to complete a domain analysis, which is a "means to determine the commonality that exists across the domain, and thus determine what types of services would have reuse value" (Rodgers, 2009). A caveat to consider in a domain analysis is the issue of service refactoring.

Employ Reuse Engineering and Reuse Engineers. For successful SOA development, it is essential to establish a position within the C4I organization that works horizontally across program boundaries. Such a position is known as a reuse engineer (Rodgers, 2009) and, besides completing a domain analysis, includes the responsibilities of:

- Consumption
  - loan to projects to assist in practice of reuse
  - communicate to programs what is potentially reusable
  - assist in adapting reusable software
- Production
  - identify what components are best candidates for reuse
  - assist in making the components reusable (Fitchman and Kemerer, 2001).

Understanding where reuse potential exists is only an initial step. PEO C4I needs to employ a reuse engineering group that has both technical and project management skills and provide it the authority and resources to work across the programs. The reuse engineers will be deeply involved in the domain analysis and use it in assisting programs in developing and using services.

Maintain Redundancy. Dependability requires redundancy in service locations on the network. However, managing multiple copies of the same service detracts from the benefits of SOA by increasing maintainability and increasing information assurance issues, but will still fit in the SOA concept (Rothenberg, 2009).

Network resilience issues pose critical challenges for the Navy in SOA adoption, but these issues can be managed through Failure Mode Effects Analysis (FMEA). Another crucial responsibility of the reuse engineer is to properly capture the network and service failure effects in a FMEA. The FMEA will guide where redundancy is essential, which capabilities are safe to publish as a service, which applications are safe for service consumption, and the proper prioritization of services and applications. The FMEA results will add another dimension to the domain analysis by supporting reuse opportunities and identifying when reuse should be forfeited though there might be a high return on investment (ROI).

Formalize a Complete Reference Model and Architecture. "A reference model is a division of functionality together with data flow between the pieces" (Bass et al., 2003). An SOA reference model is at a higher level of abstraction than a functional architecture model. One of the benefits of SOA is its agility, which will not be captured in the functional architecture model. The reference model should establish the SOA environment, which includes low-level and core services and key application services. Orchestration of these services in the reference model provides the real functionality (Rothenberg, 2010). PEO C4I should continue to develop its own reference model for SOA or adopt it from someone else. An SOA tiger team effort developed an initial "PEO C4I stack." This can be refined or reconsidered with great feedback from developers within each shop (PMW 120, 150, 140) in PEO C4I.

A reference architecture is specified by mapping a reference model onto software components and data flows between those components. Software components in an SOA are the set of available application-level and user-level services, and orchestration determines the dynamic "data flows" between these services. "Whereas a reference model divides the functionality, a reference architecture is the mapping of that functionality onto a system decomposition" (Bass et al., 2003). Users in an SOA define this mapping through orchestration. A view of this relationship is shown in Figure 5.

Figure 5
Importance of a Reference Model

RAND *MG1071-5*

Evaluate Business Process Realignment and Decomposition. PEO C4I should consider a way forward in terms of new funding mechanisms through both organizational realignment and technology guidance. A 2004 Sun Microsystems whitepaper on SOA readiness cited it as a key factor to SOA success:

> The extent to which an organization has aligned their business and technology strategies is a key determinant of SOA readiness. It is important to evaluate the business strategy for well-defined business services and processes, and to evaluate the technology strategy for separation of mission-aligned business services from enabling but nonmission-specific IT infrastructure services. (Sun Microsystems, 2004)

Two key SOA success factors listed were as follows:

> 1. Shared Services Strategy: Existence of a strategy to identify overlapping business and IT functions with the intent of reducing

> or eliminating redundancies and overlaps through use of shared services
>
> 2. Funding Model: Existence of an IT funding model aligned with and supportive of a shared services strategy. (Sun Microsystems, 2004)

Become Data-Centric. How an organization ultimately realigns can occur in a multitude of ways, but a key piece of the realignment is to excise the data from the Program Manager, Warfare (PMW), shops. In some cases, this is easier said than done, because some data reside on legacy hardware that is also managed by that same PMW, which is why the data are "owned" by them in the first place. There is a legacy cycle of dependence, in which the data are dependent on some piece of hardware, which in turn provides those data to software that has been designed to interpret the data and present them to an end user.

As noted earlier, the NTRM begins to codify the PEO into more generic functional areas. The four segments of the portfolio that are of most interest are the Common Computing Environment (CCE), Common Services, Application Services, and Cross-Cutting Services (communications and networks excluded). Agility will come to the PEO when there is clear separation of concerns, in terms of both funding and requirements for user-facing mission tools, common services and integration, legacy integration, and data management.

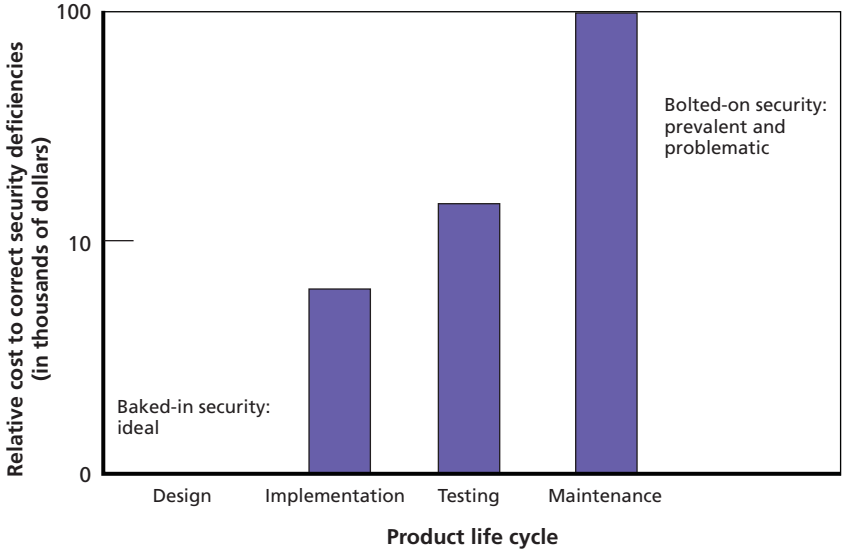
Modernize Software Engineering Practices. Many of the methods for developing software relied on within the PEO C4I are outmoded and in need of updating. Today's agile organizations benchmark themselves against industry standards of process maturity, such as the Capability Maturity Model Integration (Software Engineering Institute, 2010), to understand how well their processes are structured to allow them to respond to changing requirements and operational needs. From many of our interactions with PEO C4I, it seems that basic maturity level processes that incorporate data collection and process management are usually not followed. A common perception is that structured processes are burdensome and slow down an organization. However, we counter that ad hoc processes lack thoughtful structure and discipline, ultimately resulting in problem projects.

Track Basic Cost Information Better. Tracking basic cost information, such as earned value, is requisite to becoming a low-cost provider of services. We found that this sort of information is not currently collected in PEO C4I from the programs. Thus, progress and costs are hard to track and hard to manage.

Cost-Effectiveness

It is important to note that it is cost-effective for developers to build in (or bake in) security (information assurance) earlier in the design rather than "bolting it on" later, according to a 1981 IBM study (see Figure 6) (IBM, 1981). Therefore, it is a concern that should be addressed upfront in the development of any service-based, open architecture.

Figure 6
Cost and Benefit of Security Designed Upfront



SOURCE: Data from IBM Corporation (1981).
RAND *MG1071-6*

Final Recommendations

In order to identify those services that will assist in providing greater interoperability between systems and users at best cost, we gathered data from technical reports and interviews, which allowed us, in turn, to develop a method to identify and categorize service value as well as measure cost-related ROI. We offer the following additional recommendations for PEO C4I as it moves forward with its vision.

1. Collect the right data. PEO C4I would benefit from collecting additional data on costs and the efficiency of its developer. Current contracts do not encourage contractors to collect or share data that will help them improve effectiveness.

2. Move incrementally. PEO C4I operates in a complex environment that requires assured information promulgation, regardless of the disparate and legacy requirements, platforms, and acquisition methods that many modern organizations do not have to deal with. Trying to modernize this environment and change decades of systems engineering practices immediately would be a challenge. This incremental approach requires that legacy systems and their pairwise interfaces are maintained as the SOA environment is implemented (Rothenberg, 2010). Moreover, lessons learned from industry suggest that the use of software as services must be attempted incrementally.

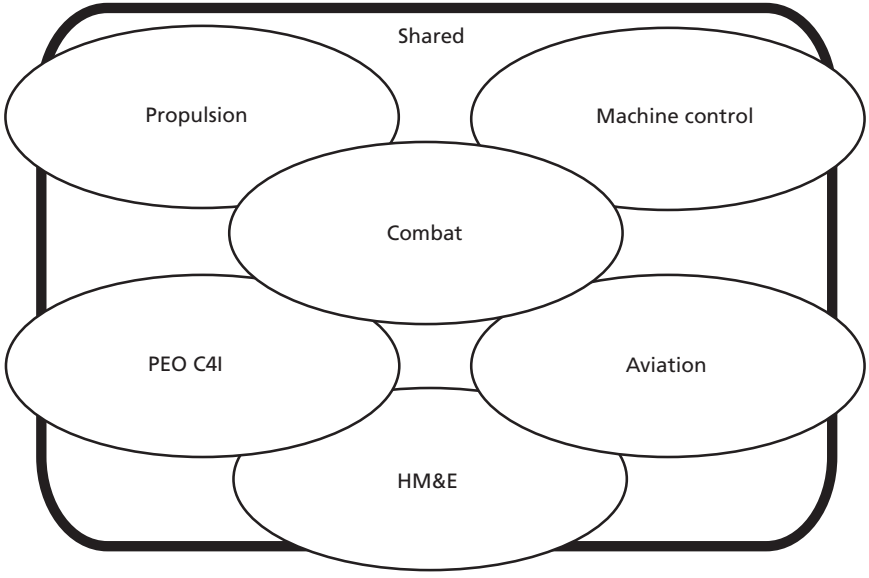
3. Consider organizational changes. Organizational change must be commensurate with an effort to transition to a service-based, open architecture for the "C4I domain." As noted by many other studies, addressing governance is paramount.[9] Furthermore, an SOA integrator is needed to act as a point person to connect business processes, data, and software in a way that makes sense for the organization. On top of it all is a need for more transparency in development that comes from maturing the organization's software engineering practices to track development cost and productivity more closely.

[9] Governance includes establishment of decision rights for the development, deployment, and management of new service, as well as monitoring and reporting decisions for communicating governance results (IBM, undated).

A Future Vision

PEO C4I provides just one of a number of networks on ships. A desirable long-term goal is a Navy-wide collection of services and perhaps an overall shared architecture that can span a ship, the entire Navy, and/or joint forces. We illustrate this vision in Figure 7, which shows the different networks on a single ship, including networks for propulsion, machinery control, combat systems (e.g., PEO Integrated Warfare Systems), aviation, and hull/machinery (hull, mechanical, and electrical [HM&E]). All of this could conceivably be integrated into one united, shared architecture. If PEO C4I is successful in utilizing services in support of an SOA for its own functional area, this will be a step toward this vision.

Figure 7
Future Vision of Consolidated Functional Area Networks



RAND *MG1071-7*

APPENDIX

When to Use Service-Oriented Architecture

It is worth noting the advice of the U.S. Army Enterprise Solutions Competency Center with regard to "when to use SOA":

- If your enterprise includes multiple stovepipes and legacy systems that have no means of communicating with each other.
- If there is no economic value in building or buying an alternative solution.
- If you want to decrease your dependency on vendor-specific software products and still use multiple software service components.
- If you are trying to maximize your ability to create flexible business processes and support cross-functional enterprise views. (U.S. Army Enterprise Solutions Competency Center, no date)

Industry and DoD Uptake

The value of SOA continues to be debated in industry, government, and academia. According to a 2008 SOA user survey by Gartner, companies considering such an approach are trying to improve and simplify access to software services and utilize a common computing environment for all of their network requirements.

Burton Group's Anne Thomas Manes (2009) wrote recently that "SOA was dead [and has] failed to deliver its promised benefits."[1] In the

[1] Manes (2009) argues, "SOA is survived by its offspring: mashups, BPM, SaaS (a superset of SOA), Cloud Computing, and all other architectural approaches that depend on services."

face of her conjecture is a Defense Science Board report (Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, 2009) that concluded, "[T]he SOA approach, under the guidance of a centralized oversight authority, offers a way to move forward with incremental acquisitions while doing so in alignment with the Department's strategic goals." Furthermore, in contrast to Manes's commentary, the Software Engineering Institute at Carnegie Mellon University concluded that "the reality is that SOA is currently the **best option available for systems integration** and leverage of **legacy systems**" (Lewis, Smith, and Kontogiannis, 2010; emphasis added).

While the concept may still be seen as potentially producing many desirable benefits, thus far, actual implementations within DoD lag the enthusiasm for its adoption. This is due in some part to the challenge of using a mix of legacy and new applications[2] (Porche et al., 2008).

How Can a Service-Oriented Architecture Be Made to Work?

Given that this report seeks to identify the existence of potential services along with their cost and benefit, it is equally important to address any limitations to fully developing a service-based, open architecture. The research team addressed a number of challenges and possible solutions that need attention before such a vision can be fully realized.

1. Become an increasingly data- and system-centric organization. Organizational limitations may reduce rapid development of an SOA's data- and system-centric environment. PEO C4I is essentially a segmented organization consisting of areas that are generally understood within the Navy and DoD, as they align similar mission competencies and have evolved working relationships to share information and objectives. This has created a legacy cycle of dependence in which the data are dependent on some piece of hardware, which is "owned"

[2] There are two camps of thought on this point. One perspective emphasizes the need to utilize legacy systems in transforming into an SOA environment. Another perspective is to develop services *ab initio* (Rothenberg, 2010).

by a PMW, and IT funding models are founded on this design. The data- and system-centric environment required by an SOA will challenge many formally clear lines of authority and funding that currently foster stovepipe systems not designed to interoperate.

2. Routinely perform domain analysis. It is well accepted that considerable upfront costs and efforts are required to realize an SOA, thus requiring the organization to function properly to gain this capability. A critical initial step for SOA realization is for an organization to complete a domain analysis to determine service commonality and reusability. This effort is the first step in a domain-wide refactoring effort in which these identified common functionalities are redefined and reallocated to achieve an SOA. It is also essential to have a position within the C4I organization that works horizontally across program boundaries to conduct domain analysis, and to employ an engineering group that has the authority as well as the technical and project management skills to optimize reuse capabilities.

3. Be aware that service redundancy is sometimes necessary. While working within an SOA will alleviate the redundancy that has hindered cost-effectiveness, there are times when managing multiple copies of the same service is essential. Understanding what capabilities are safe to publish as a service, understanding what applications are safe for service consumption, and determining proper prioritization of services and applications are important in ascertaining when redundancy is critical. Another security concern is attempts to spoof a service (Rothenberg, 2010). The results will add another dimension to the domain analysis by supporting reuse opportunities and identifying when reuse should be forfeited, though there might be a high ROI.

Though the cost-effectiveness of an SOA may be hindered by redundancy, redundancy provides additional advantages. A main motivation of SOA is the ability to use alternative services that provide differing advantages. Also, seamless integration of a new service is achieved with an SOA (Rothenberg, 2010).

4. Continue to develop a complete PEO C4I-specific reference model and architecture. PEO C4I should continue to develop or adopt a reference model, complete with data flow information, for its SOA. An SOA tiger-team effort developed an initial "PEO C4I stack," which

can be refined or reconsidered with continued feedback from developers within each shop (PMW 120, 150, 140) in PEO C4I. A reference architecture, specified by mapping a reference model on to software components and data flows between those components, may also be developed.

5. Modernize software engineering practices. Today's agile organizations must benchmark themselves against industry standards of maturity. To best respond to changing requirements and operational needs, incorporation of data collection and process management needs to be done. Specifically, tracking earned value is requisite to becoming a low-cost provider of services.

Bibliography

Bass, Len, Paul Clements, and Rick Kazman, *Software Architecture in Practice*, Pearson Education, 2003.

Department of the Navy, "N6 Agenda for Change," 2008.

Fitchman, Robert G., and Chris F. Kemerer, "Incentive Compatibility and Systematic Software Reuse," *Journal of Systems and Software*, Vol. 57, No. 1, April 27, 2001, p. 45.

IBM, SOA Governance and Service Lifecycle Management, undated. As of June 3, 2010:
http://www-01.ibm.com/software/solutions/soa/gov/

———, "Implementing Software Inspections," course notes, IBM Systems Sciences Institute, 1981.

"Market Research and Request for Information (RFI): Service Oriented Architecture in Support of the Consolidated Afloat Network and Enterprise Services (CANES) Program," no date.

Manes, Anne Thomas, "SOA Is Dead; Long Live Services," January 25, 2009. As of November 17, 2010:
http://apsblog.burtongroup.com/2009/01/soa-is-dead-long-live-services.html

Navy Program Executive Office, Command, Control, Communications, Computers, and Intelligence, *PEO C4I Master Plan, Version 3.0*, August 7, 2009.

National Research Council, *C4ISR for Future Naval Strike Groups*, Washington, D.C.: National Academies Press, 2006.

PEO C4I—*see* Navy Program Executive Office, Command, Control, Communications, Computers, and Intelligence.

Rodgers, Rich, "Reuse Engineering for SOA," IBM, September 9, 2009. As of November 17, 2010:
http://www.ibm.com/developerworks/webservices/library/ws-reuse-soa.html

Rothenberg, Jeff, "Tactical Service Oriented Architecture (SOA)," Santa Monica, Calif.: RAND Corporation, unpublished draft, 2009.

———, comments on draft of "Finding Services for an Open Architecture: A Review of Existing Applications and Programs in PEO C4I," September 16, 2010.

Schank, John F., Christopher G. Pernin, Mark V. Arena, and Susan K. Woodward, *Controlling the Cost of C4I Upgrades on Naval Ships*, Santa Monica, Calif.: RAND Corporation, M-907-NAVY, 2009. As of January 5, 2011: http://www.rand.org/pubs/monographs/MG907.html

Software Engineering Institute, "Capability Maturity Model Integration," Pittsburgh, Pa., 2010. As of November 17, 2010: http://www.sei.cmu.edu/cmmi

Sudkamp, Steven, comments on draft of "Finding Services for an Open Architecture: A Review of Existing Applications and Programs in PEO C4I," September 23, 2010.

Sun Microsystems, "Assessing Your SOA Readiness," 2004.

Tsui, Nicholas, and Dennis Shea, "The Challenge of SOA Standards Implementation in the Navy: Research and Findings," draft briefing, Center for Naval Analyses, 2009.

U.S. Army Enterprise Program Solutions Competency Center, "Service-Oriented Architecture Reference Guide," no date.

U.S. Department of Defense, *DoD Architecture Framework Version 1.5—Volume I: Definitions and Guidelines*, 2007.