Software Engineering Institute
Carnegie Mellon University

State of Practice Report: Essential Technical and Nontechnical Issues Related to Designing SoS Platform Architectures

Sholom Cohen
John Klein

May 2015

TECHNICAL REPORT
CMU/SEI-2015-TR-007

Software Solutions Division

http://www.sei.cmu.edu

Table of Contents

List of Tables

Acknowledgments

The authors are grateful to all of the government and industry experts who participated in the interviews. We also appreciate the extensive comments provided by our reviewers.

Abstract

This report presents an analysis of the state of the practice in system-of-systems (SoS) development. SoS architectures, or blueprints for integrating multiple systems based on common software platforms, have been successful in many commercial environments. The report discusses technical issues related to SoS common platform development and adoption in the Department of Defense (DoD) and the nontechnical constraints that must be satisfied. The analysis is based on information captured from 12 interviews of leading SoS developers in the DoD and industry, applying a SoS definition from the literature to identify gaps between the current state and the desired end state. The results of the study show that while commercial and DoD developers follow different approaches, all organizations report nontechnical constraints as more challenging than technical issues. For the DoD, these include leadership changes, shifting political priorities, and difficulty in replacing suppliers. The report recommends further study of SoS planning and agile approaches that better support incremental development; bridging the gap from SoS to system concerns so that system designers understand SoS concerns and can focus on their products in the context of the SoS; and documenting the platform at all software levels, including architecture views and component integration strategies.

1 Introduction

1.1 Problem Statement

When we consider the term *system of systems* (SoS), each of us summons a slightly different image. Some current examples of SoS contexts illustrate the various meanings that we apply to the term:

- large-scale business enterprise – integration of manufacturing, sales, global information, analytics, and other business operations through connecting systems across the enterprise
- small-medium enterprise – integration of warehouse, distribution, and accounting across the enterprise
- consumer or web user – (1) e-commerce solutions that link consumer marketing, sales, payment, and fulfillment and (2) mobile apps that link product search, geo-location, product availability, and customer reviews
- distributed command and control – military systems that support situational awareness, battle command, intelligence analysis, and weapon deployment and engagement
- on-board electronics (aviation, automotive, other) – integration of GPS, communications, and internet to enhance vehicle guidance and control

This report describes technical and nontechnical issues that help define the current state of practice in SoS development and operations. The context is primarily distributed command and control, with a focus on the Department of Defense (DoD) context, but we also examine the commercial, large-scale enterprise context. The report contrasts the current state of practice with goals desired by SoS developers. It also highlights issues in the DoD related to moving toward an integration platform that provides common infrastructure and applications support for SoS development.

1.2 SoS and Common Platform Definitions

A system may be defined as "a construct or collection of different elements that together produce results not obtainable by the elements alone" [INCOSE 2006]. The individual systems in a SoS must be large-scale or complex and frequently are stand-alone systems. They do not exist as a result of decomposing a SoS. Rather, each SoS constituent system has come into existence independently of the others. Given these definitions, what characteristics of a SoS distinguish it from a collection of collaborating or communicating systems?

The distinguishing factor of a SoS highlighted in this report is the issue of independence, for both operations and management of the individual elements. In many systems considered systems of systems, the individual elements may be capable of operating on their own, apart from the remainder of the SoS, but SoS development and certainly sustainment are centrally controlled. Two constituent elements may even rely on common systems or subsystems that prevent them from complete independence of operation.

The DoD recognizes that in many cases, its evolving needs and required capabilities can be met only by linking individual systems as a SoS. Dahmann and Baldwin report, “Most user capabilities require multiple systems to work together to meet user needs, so there has been increased emphasis on understanding SoS behavior toward user capability objectives” [Dahmann 2011a]. In this section, we look at a working definition for SoS and a framework to evaluate an existing or a planned SoS.

Many sources stop short of establishing both operational and managerial independence as criteria for a SoS [Clark 2008, INCOSE 2011, OUSD 2008]. Independence may have existed before the constituent systems were brought together, but the definitions do not explicitly require this independence after integration as a SoS.

- The OUSD defines a SoS in terms of a “set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities” [OUSD 2008].
- Clark discusses only the integration of the individual parts, which are themselves independently developed [Clark 2008].
- The definition of the International Council on Systems Engineering (INCOSE) does not include independence of management, stating only that “interoperating collections of component systems usually produce results unachievable by the individual systems alone.” The INCOSE definition does acknowledge “multiple, heterogeneous, distributed systems” as the constituent elements [INCOSE 2011].

A more specific definition of a SoS by Maier establishes the following qualities [Maier 1998]:

- operational independence of the elements: Constituent component systems can usefully operate independently.
- managerial independence of the elements: Constituent component systems are acquired separately and maintain an ongoing operational existence independent of the SoS.
- evolutionary development: The SoS continues to develop and evolve, through modification of existing constituent components and addition or removal of others.
- emergent behavior: Properties are emergent when the system has capabilities not resident in any component system but that emerge through interactions among constituent components.
- geographic distribution: Elements are not confined to any predetermined location or deployment scheme.

This final definition has formed the basis for the analysis performed in this report. Both researchers [Gorod 2008, Maier 2005] and practitioners [OUSD 2008] acknowledge that these properties necessitate developing and operating a SoS in different ways than in a large, complex (monolithic) system. The construction of these systems of systems also moves along a continuum from more to less central control:

- directed: The SoS is developed and operated to a common purpose, which is expressed through formal organizations, technical standards, and the socialization of its operators to the common purpose.

- acknowledged: "An organization is responsible for the SoS and supporting SoS systems engineering while independent organizations and SE [systems engineering] teams are responsible for the constituent systems that support the SoS capability objectives" [Dahmann 2011b].
- collaborative: The SoS began with a directed purpose, but now follows purposes imposed upon it by its users. Operation and development occur through the collaboration (largely voluntary) of its participants.
- virtual: No current body, voluntary or otherwise, controls all the elements. Participants (governments, corporate entities, users) will often have conflicting purposes that they will simultaneously attempt to fulfill.

The relationship of systems of systems to the concept of a "platform" also forms a key part of this report. The term *platform* refers to different concepts, depending on the context. Military vehicles, including ships, aircraft, and ground vehicles, are considered platforms in the DoD context. The concept of "platform architecture" in this context may reflect the separation of payload from platform. For example, Greenert recommends, "The design of future platforms also must take into account up front the volume, electrical power, cooling, speed, and survivability needed to effectively incorporate new payloads throughout their service lives" [Greenert 2012]. In software, the term platform is used to refer to the common elements reused across a product line, product family, or "product platform" [Cusumano 2010a]. The commercial approach to platforms that span systems from multiple organizations, termed an *industry platform* (e.g., iPhone or Facebook platforms), is described as "A foundation technology (or service) used beyond a single firm, whose value increases geometrically with (a) complementary products and services, and (b) more and more users" [Cusumano 2010b]. This definition of industry platform addresses many of the goals set in the DoD SoS concept for programs such as the U.S. Army Common Operating Environment, the U.S. Navy Open Architecture, and the multi-service Future Avionics Capability Environment (FACE).

In this report, we use the term *SoS platform* to extend the definition of the industry platform that supports individual systems from multiple organizations. A SoS platform, by extension, "must balance sufficient commonality to support economical reuse, while also providing variability and extensibility to enable innovation in system and system of systems (SoS) capabilities" [Klein 2012]. SoS platforms, like industrial platforms, provide general-purpose services, for example, directory and authentication, as well as mission-specific services, such as geospatial and message handling for command-and-control SoS. The SoS platform addresses essential architectural goals [Klein 2013]. These include

- supporting interoperation among the systems using the platform: The SoS platform provides common information models (semantics) and common communication mechanisms. The platform may also prescribe patterns or sequences of interaction for certain SoS functions.
- reducing the cost and time needed to develop or modify systems for use in the SoS: The SoS platform provides implementations of services needed by constituent systems. Those services of a constituent system that can be shared by other constituent systems of the SoS are relocated into the SoS platform. System-to-platform dependencies enable strategic reuse, which typically reduces the time and effort required to develop, integrate, and test systems to create the SoS and for an organization to create a new or replacement system. The SoS platform reduces required effort, expertise, and risk.

- enabling modular substitution of constituent systems in the SoS: The SoS platform can support creation of an ecosystem, where organizations "have a strategy to open their technology to complementors and create economic incentives (such as free or low licensing fees, or financial subsidies) for other firms to join the same 'ecosystem' and adopt the platform technology as their own" [Cusumano 2010a].

1.3 Goals of the Study

The DoD desires the ability to quickly and economically create, integrate, certify (if needed), and sustain new combinations of existing, independently developed system capabilities. The Army's Common Operating Environment (COE) and the Integrated Air and Missile Defense Battle Command System exemplify the need to rapidly field combinations of existing systems in new ways. This report contributes to the improvement of that ability through incorporation of successful industry practices.

To meet these needs of the DoD, this report answers the following questions:

1. *What processes are used to develop SoS architectures, and how do these processes use software elements in the architecture?* Current approaches do not support the quick development, integration, and certification of such system combinations or the sustainment and evolution of such system combinations after initial deployment. Conventional, top-down processes may, in some cases, give way to more incremental or agile approaches, but better methods are still needed to support such goals as evolution across the SoS and attainment of the desired quality attributes across the SoS.
2. *What challenges do SoS programs face in developing architectures; performing test, integration, and assurance; managing runtime configuration and operation; and evolving the SoS? What approaches have been used in successful programs to overcome these challenges?* Systems-of-systems activities in the DoD have had varying degrees of success. The independent management of the definition, acquisition, and evolution of the constituent systems remains a challenge and is further aggravated in the context of the SoS. Also, the external environment may undergo unanticipated changes during development and integration. These changes may apply to the constituents or to the SoS as a whole, resulting in significant change in operational needs. These challenges are partly related to the time (and resulting cost) to develop, integrate, and field SoS capabilities—longer cycles make coordination across constituent system programs more difficult and increase the likelihood that environmental changes will necessitate redirection.
3. *What are the constraints on new approaches to developing, using, and evolving these SoS architectures?* The DoD is moving toward a needs-based approach to defining new systems. Rather than define a system by specific functions or capabilities, this approach seeks to identify new and emerging operational needs and then search for or acquire systems that address those needs. In some cases, no existing system or combination of existing systems can meet those needs, but often the needs are achievable through a SoS approach, which integrates existing systems, augmented with some new development. While contributing to meet the needs of the SoS users, these constituent systems must continue to support previous users with existing capabilities.

4. *What are the important differences between practices used to create commercial SoS architectures and DoD SoS architectures?* Commercial platforms like Apple, Facebook, and Amazon have been successful in enabling the rapid creation and delivery of SoS capabilities. These platforms employ modular architectures that provide commonly used features such as identity management, geospatial location and mapping, or intersystem communications. The platform architectures include design rules and standards that allow developers to rapidly create, integrate, and deploy innovative SoS capabilities on top of the underlying common platform. The DoD has attempted to duplicate the successes of commercial platforms, with varying results (e.g., Technical Architecture Framework for Information Management [TAFIM], Defense Information Initiative Common Operation Environment [DIICOE], Managed Operation System Alliance [MOSA], Joint Technical Architecture [JTA], DoD Information Technology Standards and Profile Registry [DISR], and Future Combat Systems [FCS]). Technical barriers to success include more stringent and diverse quality attribute requirements, stronger assurance requirements, and more complex system interactions.

Addressing qualities (business, architecture, system) at the system level or across the SoS also remains a challenge. The National Research Council's *Critical Code* report notes the DoD need for effective evaluation of critical quality attributes [NRC 2010, p. 127], particularly in net-centric systems of systems, and notes that improvement is needed in the DoD's ability to manage the design, evaluation, development, and evolution of systems of systems [NRC 2010, p. 123].

SoS development and integration also face nontechnical barriers, including organizational alignment, governance, and policy [Northrop 2012]. This report will touch on these nontechnical challenges as part of the framework but will not examine them to the same degree as technical approaches.

1.4 Structure of the Report

The report uses an interview framework to analyze a number of exemplar systems of systems, primarily in the distributed command-and-control area, and address the goals of the study. As a means to explore aspects of systems that are integrated as systems of systems, the report will examine the state of practice in SoS development using the following structure:

Section 2 provides the research method in terms of the interview design for exploring systems of systems, a consistent reporting structure, and the demographics of the organizations that were interviewed.

Section 3 provides the questions and reports the results of the interviews. The interviews used the SoS definition and an interview script (contained in the appendix) as vehicles for interviewing leaders in industry and government who are developing systems of systems.

Section 4 provides results in the form of challenges and sensitivity points with recommendations for addressing them.

Section 5 provides conclusions and areas for future research for refining the interview framework, including an example for community analysis, and creating a roadmap for community collaboration in the area of SoS maturity.

2 Research Method

2.1 Interview Design

To better understand the current state of practice for development of systems of systems, the SEI conducted a series of interviews with leaders in industry and government who are developing systems of systems. These interviews covered two large-scale commercial organizations and six SoS-related DoD projects.

The interviews were structured according to the following outline. (See the appendix for the complete list of questions.)

1. interviewee demographics
2. SoS architecture development: definition, design process, architecture tradeoffs, and architecture decision process
3. SoS success patterns and challenges: case studies of success, challenges encountered, test and integration practices, and configuration management (CM) approaches
4. solution constraints: development, test and integration, CM, and sustainment

The 12 interviewees came from three categories of systems of systems. For the purposes of this report, the interview results are grouped as[1]

- commercial enterprise systems: 4
- large-scale command-and-control systems (government and industry): 4
- platform architecture development: 6

We categorized the results of the interviews to preserve anonymity and to collect common information across the various sectors.

2.2 Participants – Demographics

To obtain input for the study, the team planned a workshop—the Composing Assured System of Systems Challenge Problem Workshop—to address the need for improved approaches to SoS development. Participants—program managers (PMs), technical directors, chief architects, and other experts from the DoD and industry—would share experiences, help refine the essential elements of the need statement, and identify constraints on viable solutions. Invited participants were from the professional networks of the SEI research team members, SEI customer rolls, and DoD SoS initiative organizations. However, most invitees declined, expressing reluctance to share relevant experience in a group setting.

To overcome this objection, the team adopted a scripted interview approach—an interview protocol to capture and anonymously report responses to the questions. We re-contacted workshop invitees with direct experience as architects or systems engineering leaders on the development of at

[1] The numbers by category include interviews that covered both a command-and-control system plus the derived platform for future development.

least one SoS. Invitees were also requested to forward the invitation to other appropriately qualified members of their professional networks. This approach resulted in 14 qualified participants, 2 of whom later withdrew from the study, leaving the 12 participant interviews reported here.

We sought qualified participants with significant experience, including SoS projects and roles as architects, managers, or integrators. These qualifications were met by the 12 study participants as follows:

- years of professional experience: between 10 and 25
- participation in SoS projects: 8 with experience in four or more, 2 with direct experience in only one SoS project (2 did not provide this information)
- roles in project: architects, program or project managers, and SoS integration leads

Table 1 describes the types of organizations represented by the participants.

Table 1: Organization Types Represented

Organization Type	**Number of Participants**
Commercial software development (non-military)	4
Military system development – industry	5
Military system development – government	3

At least two researchers participated in each interview; one acted as the lead interviewer and the other researcher(s) took notes on the responses. The interview sessions followed a script guiding the lead interviewer through the questions. An interviewee's response to one question often covered several of our topics. The script served as a checklist in those situations, to ensure that all topics were covered without asking each question directly. The researchers captured interviewee responses in notes, and one researcher merged the notes into a single interview record. The lead interviewer met the interviewee in person for nine of the interviews, with the other interviewer(s) participating by telephone. The other three interviews were conducted solely by telephone. The researchers all have methodological training and experience in conducting and reporting interviews as part of exploratory research.

3 Interview Responses

3.1 Interview Questions and Groups

Questions

The four sections of the interview outline (see the appendix) helped capture specific demographic information and details about development processes, SoS success, and constraints. The question summaries from Table 2 are used to structure subsequent tables in this section that group responses. Tables 3–5 coalesce these interviewee responses to Questions 2–4.[2]

Table 2: Summary of Interview Questions

Question Section	Question Summary
1. Disclaimer and Demographics – responses covered above in Section 2.2	Provide a prepared statement to the interviewee indicating that reported results will be summarized to maintain anonymity and protect the privacy of the interviewee and the organization. The lead interviewer also asked the interviewee not to disclose proprietary information that could not be released even if the identity of the interviewee and organization were anonymized. Demographic information included the years of experience of the interviewee, years of specific SoS experience, and numbers and names of SoS projects and programs.
2. SoS Architecture Development – responses covered in Section 3.2	a. Focus on the processes used to develop SoS architectures: • interviewee definition of the term *system of systems* • general approach to the architecture design (not documentation) process • bridge from SoS-level to system-level focus • software treated as part of or separate from rest of SoS • hierarchical vs. layered decomposition b. Focus on architecture tradeoffs • how software concerns interact with other SoS concerns • how to balance constituent system needs in SoS context • framing architecture tradeoffs and decision making • factors (technical and nontechnical) influencing decisions
3. SoS Success Patterns and Challenges with reference to a specific system, SoS, or program to identify specific gaps in current practice where new methods would have the greatest impact and to identify specific solutions employed by the interviewees that would be candidates for generalization – responses covered in Section 3.3	a. On development of constituent systems for the SoS (with examples and evidence of success) and of SoS-related challenges in the development of constituent systems for use in a SoS b. Success and challenges (technical and nontechnical with examples and evidence of success) in test, integration, and assurance of constituent systems in the SoS c. Success and challenges (technical and nontechnical with examples and evidence of success) in runtime configuration and management of the SoS d. Success and challenges (technical and nontechnical with examples and evidence of success) in sustainment and evolution of the SoS
4. Solution Constraints on a new method for design or analysis to address the challenges –	The final set of questions focused on the constraints that a solution must satisfy. Focusing on the same four activity areas used in the previous question set, identify factors necessary for any new approach to be successfully

[2] In this report, references to actual DoD or industry developers or developments have been made anonymous, to the extent possible, to protect confidentiality of information while preserving the information captured from the interviews.

Question Section	Question Summary
responses covered in Section 3.4	translated from research into practice. Constraints can be technical, organizational, governance, policy, regulatory, incentive, doctrine, or other cause.

Interviewee Groups

The SoS characterizations in this section are based on the interviews conducted with programs from each of the three groups. The information from across SoS groups has been integrated to provide general descriptions without revealing specifics of any actual system or development.

Commercial enterprise systems

The interviewees characterized these systems by "what the SoS platform does": SoS will support the business processes that create and use transactional data as well as support applications accessing the customer's data for nontransactional uses such as analytics. Organizations that develop and deliver such integrated applications most often build these systems of systems using commercial platforms. Each platform targets a broad segment of the mission-critical business system market, delivered in an internet-scale "platform as a service" model [Mell 2011], and is capable of supporting an ecosystem.

Large-scale command-and-control systems (government and industry)

The interviewees characterized these systems as providing integrated support for a variety of warfighter processes. They may cover shipboard, ground-based, or airborne systems and may support interoperations among two or more of these categories. In practice, the DoD interviewees consider these systems to be a type of SoS combining some number of sensor, intelligence, command, and weapons capabilities, all of which are provided by systems in their own right, through communication channels [Frank 2008, McConnell 2010]. The individual systems may be built to work together and therefore do not strictly qualify as composing a SoS in the Meier definition, since they do not exhibit independence of operation. However, some elements may be built and sustained independently.

Platform architecture development in the DoD

Platform support addresses a variety of mission areas: avionics, large-scale simulation systems, and command and control. Across these areas, platform support comes in many forms, from abstract models to direct implementations. When systematically reusing platform application models as building blocks, elements of the SoS can immediately share information or other services. Implementation of applications beyond the interfaces remains a system-specific activity, for the most part. In contrast to the commercial enterprise SoS characterized by delivered capabilities—"what it does"—DoD developments are characterized primarily by constituent elements—"what it is." They provide middleware—common software that all SoS users must link to for infrastructure across the SoS—to support integration of independently managed and operated systems. An application programming interface (API) makes messaging, object update, and other services available. The infrastructure may be extended for individual system support at the application layer through custom development or through shareable platform development. While infrastructure layer support has been achieved, another objective of these platforms has been to achieve strategic reuse of infrastructure-layer software by SoS developers at the application layer to provide a variety of goals: shorter time to field, higher quality, increased user familiarity with related systems, and ease of integration. However, to date, these developments have not demonstrated strategic reuse at the applications layer.

3.2 Question Section 2: SoS Architecture Development

In this and the next two subsections, interview responses are organized according to questions in Question Sections 2, 3, and 4 from Table 2. When a specific system in one of the three categories is cited, the phrase “in one case” is used. General responses are provided first, and a table provides contrasting answers to the questions.

Most participants from the three organization types identified two development scenarios: creation of a new SoS composed primarily of new constituent systems or integration of existing systems to create a SoS. The first scenario applies primarily to directed systems of systems, in which constituent systems goals and governance are aligned well with those of the SoS [Maier 1998]. In this scenario, architecture design can begin either top-down, based on requirements with a platform emerging as the design matures, or bottom-up, creating a platform first and then defining systems that use the platform. In the second scenario, SoS architecture is constrained by the need to integrate across diverse constituent systems. Consistency or conceptual integrity across the SoS, as needed to develop a platform or even to efficiently sustain the SoS, may not be achievable without substantial rework of constituents (and hence additional cost).

The variety of underlying system concepts results in many different approaches to address architecture drivers. Is a system with multi-mega lines of code (LOC) for a single entity under centralized development a SoS, or should it be considered a large single system? For the DoD, software systems of a single aircraft, ship, vehicle, or command post may be regarded as a SoS. Large-scale integration support efforts, such as Victory for vehicles and Navy Open Systems, are targeted to SoS support. In other cases, the focus is on a bigger SoS—for example, multiple ships or ship, aircraft, and ground-based systems in a cooperative engagement capability. The latter type is more similar to commercial enterprise systems in linking diverse computing capabilities not initially built to interoperate. In either case, a SoS development may proceed as a clean-sheet, top-down development, or the SoS may integrate existing systems, possibly with minimal modification.

Table 3: *Comparison of Answers to Questions in Section 2 – SoS Architecture Development*

Questions	Commercial SoS	DoD SoS	DoD Platform
2.a. Processes Definition of the term *system of systems* General approach to the architecture design (not documentation) process Bridge from SoS-level to system-level focus Software treated as part of or separate from rest of SoS Hierarchical vs. layered decomposition	Success depended on proper consideration of future needs, defined through market analysis. Organizations rely on deep domain experience to determine platform scope decisions. In one case, the initial platform architecture and implementation were factored out of an existing application suite. In another, the platform was a new development effort in a market space where the organization is one of the market leaders. Both organizations valued acquiring market share higher than optimizing any technical or business metric. Elements may be introduced by users of the platform in response to a corporate or mission need that was not anticipated as part of the original development.	Software was not an early concern—it was initially treated like any other element of the SoS architecture, but as the architecture design matured, concerns such as maximizing software development efficiency, minimizing development cost, and meeting development schedules were high priorities that were balanced against overall SoS measures of performance. In a clean-sheet design, the solution may start as top-down decomposition. Developers understand and conform to a basic architecture approach. When the solution is based on existing systems, no consistent architecture approach exists across the components.	Success depended on proper consideration of future needs, informed by a science and technology investment roadmap, to overcome lock-in through nonproprietary platform solutions. May use large-scale legacy systems as a starting point or may mine for SoS support. Resulting platform architecture contains elements of the infrastructure generalized to address future SoS needs. Limitations include • lack of SoS architecture for integration with other SoS products • addressing 60 percent lifecycle cost of sustainment exacerbated by contractor lock-in when elements reach end-of-life obsolescence • aligning platform ecosystem output with program schedule needs • existing systems engineering processes do not meet high-level objectives for SoS platforms To address the lack of off-the-shelf methods or tools in one case, platform developers synthesized key process areas to meet the highest level goals: affordability and time to field.
2.b. Architecture Tradeoff How software concerns interact with other SoS concerns How to balance constituent system needs in SoS context	Time to market was the primary decision driver. After a viable solution was identified, there was little additional solution space exploration. Decisions framed in a context that included both functional and quality attribute requirements. They did not explicitly distinguish between the two types of requirements.	Downstream lifecycle costs and sustainment costs were less important than SoS operational performance. Extensive trade studies were performed, with architecture decisions frequently driven by development constraints. The way software architecture concerns are framed has changed over time. Earlier projects framed decisions only in terms of functional requirements, while more recent projects are framing decisions in terms of both functional and quality attribute requirements.	Organizations recognize that it is not possible to address affordability, risk, or other concerns all at once. The stakeholders collaborate on decisions that will affect all SoS systems. Each PM implements the changes independently, and integrations are tested. To address the tradeoffs, • define components and interface standards, and allow technology innovation within those boundaries This has been labeled “modular innovation.”

Questions	Commercial SoS	DoD SoS	DoD Platform
Framing architecture tradeoffs and decision making Factors (technical and nontechnical) influencing decisions		Many constraints with the individual systems make it hard to maintain architecture "purity." Those existing systems are not changeable and the solution lives with what's there, as is. It may only be possible to manage interfaces.	• gather or synthesize data about the as-is architecture and technology to identify areas for cross-cutting innovation • modularize to raise levels of reuse; document at all software levels (architecture views) • assure future platform users that conformance will also address performance and that funding to get there exists • base investments on middleware or on platform feedback

3.3 Question Section 3: SoS Success Patterns and Challenges

Software product line and platform-based engineering practices also promote the reuse of assets other than software, such as tools, plans, templates, test equipment, test cases, and personnel training and skills. Architects of military SoS platforms included assets such as documentation, training materials, and user community collaboration repositories as part of their SoS platforms.

Nontechnical factors, rather than technology, dominate the challenges in developing and evolving SoS architectures. These factors include

- misalignment of development organization and authority with the architecture
- misalignment of system and SoS goals
- reluctance to introduce dependence on the SoS platform into the constituent system architectures
- regulatory and policy constraints (for systems acquired by the U.S. government) that diminish the potential value of a SoS platform approach
- need to change the procurement approach, separating capability development from individual aircraft, simulation, or other product development

These reported challenges are similar to those of developing, adopting, and sustaining software product lines [Northrop 2012]. Experience from software product lines and platform-based engineering provides insight into some of the challenges of platform-based systems of systems. Practices that are successful for single products need to change to achieve success in a product line context. Similarly, practices focused on developing single systems must change to be successful in the context of a platform-based SoS. Practices used for software product lines consider the relationships among development, organizational, and management concerns and recognize that architecture and technology are just two contributors to overall product line success.

When developing or evolving systems to use the SoS platform, many participants reported challenges related to documentation of the constituent systems within the SoS. Although extensive architecture and design documentation may exist for a constituent system, it is often focused on the independent operation of the constituent system and does not adequately address concerns related to the constituent system's operation in the SoS. Examples included resource scheduling approaches and handling of interface errors or exceptions.

The large scale and complexity of the SoS architecture context create several challenges for the initial instantiation of the SoS platform. In the DoD, platform development shifts responsibilities from individual PMs to an open government or industry organization. Maintaining backward compatibility for systems using the SoS platform was reported as a challenge in architecture evolution.

Table 4: Comparison of Answers to Questions in Section 3 – Success Patterns and Challenges

Questions	Commercial SoS	DoD SoS	DoD Platform
3.a. Development On development of constituent systems for the SoS (with examples and evidence of success) and of SoS-related challenges in the development of constituent systems for use in a SoS	The commercial SoS platform architects reported several approaches to creating and delivering the initial instantiation of the platform. From these, we have identified two "proto patterns" [Wiki 2009]: an architecture evolution proto pattern for evolving a new platform and a proto pattern for deploying new platform features. These and other patterns are reported in Section 5.	Developers follow a classical systems engineering approach. They decompose requirements from Operational Requirements or Capabilities Description Documents (e.g., capabilities, endurance, number of operators, ordinance, and communications) to technical system, operator, safety, and security requirements. A final decomposition allocates segment, element, component, or configuration items (CIs) to computer software components. Architects use the "V-model" [Forsberg 1991] to develop their systems of systems. The relatively long period between architecture definition and system integration allows some architecture errors to remain undiscovered until late in the development process. One product addressed this challenge by shifting to an iterative agile approach during later phases of the development cycle. This enabled faster feedback on the correctness of design decisions, but unresolved questions remained: Is it practical to use an iterative approach from the beginning of the project? Should an initial base of functionality be in place before beginning an iterative approach? What is the best way to plan iteration contents and duration?	Focus on environment rather than product contributes to a platform that encompasses development and testing considerations. The platform architecture defines logical interfaces, user relationships, testability of components to achieve one-time test at the component level, and support for integrated component testing at the system and SoS levels for any configuration. This platform development is seen as a way to achieve affordability. Another part of the platform environment focuses on a software tool chain—identifying tools to be provided as Government Furnished Equipment (GFE) to complement the platform. For one interviewee program, this is not yet a specific toolkit but may include specific data modeling tools.
3.b. Test, integration, and assurance Success and challenges (technical and nontechnical with examples and evidence of success) in test, integration, and assurance of constituent systems in the SoS	Nearly all testing was automated, with one organization reporting that it had "tens of thousands" of automated tests, which allowed it to maintain full compatibility back to systems developed for the first versions of the platform (the platform is now almost 10 years old and is updated three times per year).	Agile approaches test using stimulator/simulator at all code levels before final integration. They also use a test automation strategy to cover syntax and semantics of interfaces. Modeling and simulation become part of the test environment to extend test coverage beyond platform interfaces. Systems integrator identifies near-term capability testing and downstream full-scale test events, then develops simulations to play test	Currently, no process exists for one-time testing, so that quality remains a long-term objective. To advance new certification processes, one platform development funds the certification authority to work with the architecture team. Platform developers set information assurance (IA) certification improvements for constituent systems/applications. This assumes that middleware releases are aligned with those certification plans.

Questions	Commercial SoS	DoD SoS	DoD Platform
		data through applications during test, before integration. High-level architecture and closed-loop SoS simulations may be used during test and integration. As new systems are introduced, the risks of retiring operational, top-level programs must be identified. The SoS organization must develop integration and testing capabilities to assure that new and previous systems can still operate within the SoS. A significant challenge is to maintain the budget in light of these changes and maximize test efficiency to meet schedules. Most developments invest to some degree in test automation to overcome limitations of systems previously tested with manual integration processes. Developers also see automation as achieving conformance as a tested quality rather than being a "best professional judgment" issue. Organizations that develop and deliver integrated capabilities build and integrate these using a common infrastructure [OPEN 2013, Raytheon 2006, U.S. Army 2012] or integration platform. Each infrastructure targets a broad segment of mission-critical system area, generally as defined by a specific DoD service. These offer potential as common platforms for multiple system or SoS development.	
3.c. Runtime configuration and management Success and challenges (technical and nontechnical with examples and evidence of success) in runtime configuration and management of the SoS	Organizations instrument their platforms to provide visibility into how platform features and services are used by customers. The instrumentation supports both reactive activities, such as help desk support, as well as proactive activities, such as optimizing commonly used transactions within the platform.	Focus is on runtime environment rather than product. This focus may still lead to systems under control of a single integrator for both infrastructure and most constituent systems but also offers a path to a SoS platform. In some cases, as many as a dozen systems that need to interact are integrated in this fashion. When new capabilities are identified—for example, a fused track assembled from multiple sensor tracks—multiple stakeholders with systems	For continuous runtime support, the organizations behind platform development recognize the importance of a diverse inner team—all with background in the mission area and specific expertise as systems architects, systems engineers, modelers, SoS architects, and standards experts, with reach back to others working on as-is systems. An outer team includes the usual stakeholders such as representatives from a program executive office or equivalent governance body, current

Questions	Commercial SoS	DoD SoS	DoD Platform
		architected for autonomous operations must collaborate. This level of collaboration requires agreement on mechanisms for integration of existing systems, possibly based on a pub-sub or other information exchange, built into the infrastructure.	and future system PMs, and product managers (in some cases, from industry).
3.d. Sustainment and evolution Success and challenges (technical and nontechnical with examples and evidence of success) in sustainment and evolution of the SoS	New platform features frequently duplicate existing features provided by existing applications within the SoS. An organization wants applications to use the platform-provided version of the feature and promotes the value propositions that the short-term expense of migrating to the platform-provided version of the feature will be offset in the long term by increased stability and scalability resulting from use in multiple systems and configurations and that integrating with the platform will provide lower cost access to other desirable features, such as analytics or security features. The organization prefers this long-term value proposition approach as an incentive to motivate migration, rather than providing direct economic incentives (e.g., subsidies) or punitive actions (e.g., prohibiting applications that do not use certain platform features).	Also encounters challenge of justifying short-term cost of migrating constituent systems to use of SoS platform. Military organizations tend to rely on top-down mandates or direct funding incentives. Command-and-control SoS architects reported issues in maintaining compatibility among constituent systems as the architecture underwent evolution throughout the initial SoS development iterations. The DoD has made frequent attempts to expand the infrastructure through applications as programs evolve, but this has proven harder to accomplish due to programmatic constraints—on the programmatic level, this involves expanding up, but too many decisions had already been made in these structural elements, with specifications set, that were in conflict. Data modeling has proven one area of common influence across multiple systems in a SoS.	IA certification and, possibly, mission assurance certification may serve as incentives to encourage air-, ship-, or ground-based systems of systems to conform to use of the infrastructure. The emphasis is to shift from standalone legacy components to conformance via the platform. In one case, the ecosystem is organized under the auspices of an international standards body, with participation from across all services, industry, and academia. Technology roadmaps address affordability during sustainment by breaking unique hardware constraints, working with industry and subject-matter experts in other services, and creating consensus-based open standards.

3.4 Question Section 4: Solution Constraints

Participants identified constraints that must be satisfied by any new approaches or methods to address patterns and challenges discussed in the previous section. These may be generalized as a need either to integrate with existing tools or to align with assurance or certification processes.

The first constraint is that any new approach or method should be integrated with existing tools, including tools used for architecture modeling, analysis, and documentation and tools used for project or program management. The need for integration with architecture tools was expected—adoption of new approaches and methods is facilitated if there is no need for acquiring or learning new tools. The need for integration with project or program management tools indicates the strategic importance of architecture decisions; the necessity of efficiently translating decisions about technical approaches into cost, schedule, and other metrics relevant to program executives; and the significance of reflecting program constraints as architecture drivers.

The second constraint was that any new approach or method must align with assurance and certification processes. A significant benefit of a SoS platform lies in reducing the cost and time to perform assurance and certification of the SoS, but this benefit likely can be accrued only if the features included in the platform, the analysis of the platform architecture, and documentation provided for the platform are aligned with the assurance and certification requirements of the SoS.

In the DoD context, acquisition policy, guidance, and governance pose at least a perceived constraint. Developments are tied to programs. The acquisition of an independent platform that spans programs is a stated DoD goal. But the ecosystem to use and sustain the platform has not materialized on its own and is not incorporated into current program policy and governance. These policies limit the ability of programs to create effective incentives for acquirers and suppliers to join or participate in an ecosystem. Political priorities, leadership changes, and contractor relationships add additional constraints.

The responses in this question section did not exhibit the same degree of variety as those in the other sections, hence the sparse nature of the matrix.

Table 5: Comparison of Answers to Questions in Section 4 – Solution Constraints

Questions	Commercial SoS	DoD SoS	DoD Platform
4.a. Development On development of constituent systems for the SoS (with examples and evidence of success) and of SoS-related challenges in the development of constituent systems for use in a SoS	Define a platform in terms of creating network effects that support an ecosystem rather than in terms of technology characteristics such as APIs and programming languages or in terms of services provided.	Key drivers of the SoS are development constraints: maximize development efficiency, minimize development cost, and achieve schedule. Balancing SoS performance requirements against constraints is a common theme, and full lifecycle engineering through sustainment is a lower priority. Constraints on creating an ecosystem: • must reduce SoS acquisition costs by enabling modular substitution of SoS elements and creating competition among suppliers of the elements • need for increased innovation from a broader community of contributors to new and improved SoS capabilities • U.S. government acquisition policies that limit ability to create effective incentives for acquirers and suppliers to join or participate in the ecosystem	Roadmaps include both technical and nontechnical drivers, with technical roadmaps regarded as the easy part. Obtaining mandatory conformance to the architecture from PMs and reflecting that mandate in roadmaps are the primary nontechnical challenges. Once conformance is the standard, organizations can use the roadmap milestones to estimate costs and the ability to meet long-term needs. Creation of shared resources for use of the platform. These resources include training courses that provide information about the platform and its use. Others include repositories or Wiki support.
4.b. Test, integration, and assurance Success and challenges	(No responses in this category)	(No responses in this category)	Platforms leverage software other than code within middleware elements. Can no longer use traditional LOC metrics to measure progress—more integration and test with platform than traditional code and test.
4.c. Runtime configuration and management Success and challenges	(No responses in this category)	Developers establish specific design constraints for these CIs. An example is certain message classes that are safety critical and must use a specific common service. In contrast to what we found in the enterprise platforms, SoS infrastructures tend to be defined via standards in terms of technology characteristics such as APIs and programming languages. Similarly, connections within the SoS and to other existing programs are accomplished via interface requirements. Implementers have very little freedom—application developers are given requirements and interface specs to maintain consistency across the SoS.	(No responses in this category)

Questions	Commercial SoS	DoD SoS	DoD Platform
4.d. Sustainment and evolution Success and challenges	These platforms should support two-sided markets where user groups provide each other with platform benefits [Rochet 2006]. At the heart of the platforms is a repository that stores a platform customer's business data and provides support for the business processes mentioned above.	Software development is constrained by system perspective that software is implementation, so decisions are often deferred; this can have impact on the overall architecture. Also, working across existing systems requires a higher degree of cross-organization cooperation, on both the customer and supplier sides. Even when individual systems start from a clean slate, maintaining the vision and consistency in approach throughout the development is a challenge to all developers, especially as cost and schedule become primary drivers. DoD systems of systems tend to be directed or acknowledged—most of the constituent systems are developed or redeveloped for a specific SoS. They are rarely greenfield designs but integrate legacy systems that may go through parallel reengineering processes to bring them into alignment. Alternatively, reengineering may take place during planned upgrades to legacy systems. As part of the reengineering, developers will balance commonality for reuse of infrastructure services and specific needs. Rather than an ecosystem, stakeholder representatives are under one primary contractor for the complete SoS with authority over the work of an integrated product team.	Platform developments may be constrained to address existing system program "decision points" so that technology roadmaps align with those points. While software reuse at the mission processor level across mission systems is achievable, the reality is that Programs of Record (PoRs) and PMs see the systems they control as unique, and unique approaches are still considered the path to performance and size/mass reduction. Users and prime developers collaborate on revision, versions, version checking, and regression testing across systems and during release. Identify emergent behavior and determine whether innovation through composition can provide new, unanticipated capabilities.

4 Results

This section identifies challenges and sensitivity points that the DoD should address to successfully develop systems of systems using platform-based approaches. One mechanism to address the gaps could be the gradual movement of PoR systems within DoD shipboard, ground-based, or airborne systems to SoS platform approaches. However, even within platform developments, legacy and near-term product requirements may challenge management, operations, and environment-vs.-product independence.

Ties to existing programs may impede creation of platforms or other solutions that are in a position to address emergent behavior. For example, dealing with a new threat may require the ability to integrate multiple sensor streams across constituent components. The sensor streams are within scope of the SoS, and each component could build a real-time integrated view from existing and new streams, which is the desired emergent capability. But the specific components that own the streams currently process them and then can only distribute data via messaging rather than in real time. As an alternative, platform-based services could accept and distribute the streams as required by the individual components for the real-time views. The current message-based limitation imposed by processing at the component level, or similar limitations on addressing desired emergent behavior, comes from the legacy constraints of many SoS projects. Addressing the breadth of new requirements means building in longer term flexibility either to make changes at the component level to satisfy emergent needs or to build platforms that support the desired behavior and then integrate components to take advantage of support provided by the platform.

This section addresses challenges in the following areas, divided into technical and nontechnical concerns, following SoS issues raised by Maier [Maier 1998] and others identified in the interview process:

- dependence on central management and operations
- evolving needs and sustainment
- emergent behavior
- binding decisions to physical sea, ground, or airborne systems
- assurance and certification needs

The accumulated comments from interviews are encapsulated in the coverage of these areas.

4.1 Dependency

Independence for the SoS in operation and management requires constituent system strategies that are not currently in sync with most DoD program strategies. True independence requires separately acquired and sustained constituent systems. In operation, these constituent systems are not dependent on each other for needed capabilities—they continue "to operate to fulfill [their individual] purposes if disassembled from the overall system" [Maier 1998] but will rely on platform capabilities. When integrated, each constituent contributes its own capabilities to the operation of the SoS. To achieve independence, SoS planners in the DoD must find ways to overcome current procurement approaches of constituent systems that tie them to a resulting SoS.

Technical Concerns

While programmatic challenges dominate in the system area, technical methods established for DoD systems can scale up to meet these demands for the SoS. Current approaches tend to build in dependencies to address budgetary, schedule, or other constraints. To overcome these constraints, the platform approach should build into the platform those capabilities on which multiple systems are likely to depend, but this requires early attention to SoS-wide concerns captured in early system artifacts. The DoD appears to have made progress at the individual system level. It will now have to apply this thinking to the SoS.

Across constituent systems, developers already attempt to identify commonalities and isolate these via components in appropriate architecture layers. Among these components may be services such as runtime startup, register, close out, and other functional identification capabilities. Scaling this to the scope of the SoS will factor out dependencies to the platform, allow greater independence among constituent systems, and allow a single point of sustainment and evolution for platform dependencies. While reuse may be a factor outside of the infrastructure layer, the application layer reuse is usually of the clone-and-own variety, perpetuating some degree of dependence.

While common technology or standards receive attention, interface interactions and impact on systems at either end of that interface require at least as much attention. Performance modeling and simulation can assess the impact of proposed design decisions early in the lifecycle and avoid costly modifications during SoS integration. The use of simulations as an element of integration and test (see Question 3.b in Table 4) demonstrates that simulation and modeling can be applied across the development lifecycle. The “sim-stim” approach provides another check on dependencies that might affect performance or even correctness of constituent systems once integrated and can deliver schedule and cost savings.

Agile development and continuous integration may assist in this arena through frequent sharing of constituent system component developments, rather than relying on late-stage integration. Regularly performed trade studies with options can also support cross-system work within the SoS.

Nontechnical Concerns

Addressing the technical concerns of the SoS will require non-program-specific funding. This funding will address commonality and built-in variations at the level of cross-constituent SoS systems or SoS-wide concerns such as the platform. Reserve funding that management holds for contingency planning or targeted budgeting would be needed to support these non-program-specific efforts. These efforts have been attempted, especially in the large-scale training area, where common architecture and implementation efforts have provided support for multiple programs. Again, success in addressing system-level concerns could be broadened to the SoS.

Multiple management and funding streams for systems that must interoperate are another non-technical concern. The success of the SoS requires balancing the priorities and goals both on the customer and supplier sides. SoS planning must build anticipated changes into funding profiles and system needs, to avoid excessive SoS replanning and restructuring when aligning new plans across organizations.

One difficulty that programs encounter is diminishing adherence to long-term goal setting and maintenance as a program matures. In early stages, maintaining the SoS vision is easier to accomplish than later when cost and schedule become stronger priorities for individual systems. Large-system developments have benefited from maintaining a cross-SoS team to address high-priority or essential integration concerns with authority and focus on big architecture issues. This approach requires "surrender" of some autonomy by individual system development teams to broader SoS needs, but it may be necessary to maintain the independence of those systems that will collaborate in the SoS and retain integrability. To sustain the drive to meet initial program goals, programs must demonstrate the value of the cross-SoS approach early on, maintaining the long-term vision as the SoS evolves.

4.2 Evolution and Sustainment

With the SoS in transition to greater independence among constituent systems, the nature of evolution and sustainment will change. The architecture drivers for many older, large-scale systems included constraints such as minimizing memory footprint or dealing with proprietary hardware. These specific constraints are generally not valid today or are less significant with use of commercial hardware and, for non-mobile systems, availability of virtual memory and server-based architectures. Over the years, different parts of such systems have evolved in different directions, for example, toward openness or toward ease in adding new capability, and drivers reflect those system goals. Even in current systems that have achieved openness or ease in evolution, different approaches to these drivers may become a part of long-term sustainment. In addition, the need for enhanced connectedness has resulted in increased complexity, and previous methods may not scale.

Technical Concerns

Across a related set of system products, or product line, a common source code repository with branches may exist for the entire product line. Configuration management (CM) accounts for variations across the products or systems. However, this approach is not currently applied to deal with multiple versions of releases. For example, the subsystem elements of all ships in a product line will be managed through CM at the subsystem level. But the individual tracking systems across a set of ships, though closely related, will not each evolve under the same CM. Within these legacy systems are many distinct baselines that have originated due to differences in ship equipment. Adding to the current level of complexity is the scheduling of rollout for a release to the entire fleet—there is overlap in deployed baselines. The challenge is that a fix may apply in different ways to multiple configurations.

Still another factor in evolving the existing breed of large-scale systems is the need to interoperate with externally provided Government Furnished Equipment (GFE). These GFE products go through their own major evolutions and only afterward does the large-scale system have visibility into the changes that are coming. The organization must determine whether to extend itself to address interoperation with the GFE through evolution or to rebuild the affected parts.

A similar paradigm exists to integrate SoS to weapons platforms. The design of SoS software platforms should permit ease of system integration over current approaches by isolating changes that occur in a specific weapon platform and by creating a common means of SoS interoperation with the weapon platform. In some cases, interoperation enabled by having a SoS platform simplifies the current demands of system-to-system integrations in the absence of such a SoS platform. The systems integrate through the platform, a many-to-one problem, in contrast to direct system-to-system integration, a many-to-many problem.

While platform approaches cannot address all of these issues, they are seen as primary drivers. Model-based methods as part of the platform are also seen as useful tools for evolution, if models are developed and maintained over time. Models aid traceability to gauge the impact of internal system changes or of changes brought about by external interoperations. Maintaining models gives programs the ability to easily perform impact analysis when change is requested and can provide guidance to external developments to direct their implementation. It can also support upgrades that are outside the control of the system. Knowledge of change and predictions of impact may also help limit the coverage of test events to specific system activities, not requiring total system retest. Without this approach, the impact on testing is hard to predict.

Product variations may also be accommodated in the platform. For example, a product line of ship systems will vary by different collections of sensor, weapons, command systems, and human–machine interfaces. The platform can build these into its models and provide the architecture or even components and interfaces across the product line. Once systems are fielded, upgrades may proceed between release cycles to both the shared platform and the individual systems. While maintaining test alignment and interoperability is still a challenge, the platform affords more flexibility in evolving each system and minimizing the impact of change across the product line.

Nontechnical Concerns

Scaling from the system to the SoS involves issues that are more programmatic than technical. These include managing interfaces across systems, planning and executing integrated test events, and sustaining the underlying infrastructure. For system independence across the SoS, the constituent systems must have operations-level agreements to make sure that the needs of one system that must come from an external source are provided by another in the SoS, that test engagements can be coordinated, and that all elements can share access to data to build and compare impact analysis. Tooling must include SoS-wide data modeling and interface management tools along with common infrastructure tools to coordinate communications, data management, and security across the SoS.

4.3 Emergent Behavior

At the system level, particularly in a directed or acknowledged SoS, the organization views emergent behavior as providing a means to deal with new requirements within the SoS. The emergent behavior can be realized through integrating capabilities across component systems or by utilizing platform capabilities in combination with those of the constituent elements. The identification of new capabilities and the ability to address new requirements are part of engineering change. The collaborative or virtual SoS should be designed to recognize the potential for new capabilities emerging from those already built and even respond to an unanticipated threat. The capabilities of the constituent components support a breadth of scope within SoS projects that can be exploited to address unanticipated requirements. Such innovation involves weighing the risk of investing in a potential that may never be realized (the ability to easily exploit emergent behavior through architecture or other means) against the opportunity that emergent behavior can support.

Technical Concerns

The architecture of the SoS platform should look forward to anticipate technology discontinuities. The method used to reduce or eliminate dependencies should support incorporation of new behavior without significant change to individual systems and certainly with minimal change to the platform. Successful strides in SoS platform development to address the potential for emergent behavior include the ability to rapidly create exploratory prototypes. These prototypes can demonstrate innovative capability that can be fed back into mainline development. The platform can enable SoS benefits from specific categories of emergent behavior:

- operations across new networks with existing or new data exchanges
- interoperations with new systems supported by interoperability tactics (e.g., capability discovery or workflow management) within the SoS architecture
- identification of innovative capabilities that exploit the ease of harvesting design and code and building into the baseline

4.4 Deployment and Binding Decisions to Physical Systems

Leaders still think about physical products (tanks, ships, aircraft, etc.), and their perspective of SoS tends to be hardware oriented. However, a greater degree of software awareness is emerging through the leadership ranks along with recognition of the need for engineering discipline and the need to enforce that discipline on the process for developing software. The moves to complex software systems and to the SoS have engendered a refocus on the software running on these products. These leaders understand the commercial example of a Google search engine running on a desktop, laptop, tablet, or mobile product as the same software on a variety of physical products. They want to achieve the ability to field a limited number of instances or versions of common software with ease of test and integration and to avoid major cost impacts. But they still perceive development as targeted to a specific physical system.

Technical Concerns

The integration of systems into a SoS exposes fragility in system designs and deployment issues that may not be as apparent in stand-alone operation. Release of capabilities into systems and the interactions between programs to realize SoS capabilities require coordination—of the data model semantics, data link quality of service, and other decisions. The complexity of coordinating among multiple systems and suppliers has, in the past, led to a sole-source approach for the life of a system. An open infrastructure with multiple suppliers requires an architecture approach that supports late binding of the implementation to the physical product and independence among individual systems. Ideally, this approach limits the role of the system integrator to final deployment to avoid dependence on the experience of a single company throughout the development.

Nontechnical Concerns

The DoD is not perceived as having market influences such as the need to respond to competition. However, the DoD does recognize the need to separate or isolate deployment decisions from the development. This change is reflected in the goal of limiting or eliminating contractor lock-in, lengthy test and integration cycles, and costly sustainment. The common platform approach that can separate deployment from development makes buy-in at the highest levels critical, since there is no easy objective measure of strategy correctness. Such change is not well received without concrete ways to defend or justify the change. Compared to the commercial context where showing bottom-line results is feasible, the picture in the DoD requires careful management to avoid the perception that such change is a threat.

4.5 Information Assurance and Certification

For many legacy systems, even large-scale systems, information assurance (IA) was not a requirement at the outset. IA is being added as a “new” requirement now, even where security was not a factor in the original architecture. Even where IA and other safety or security requirements are part of a system specification, acquisition processes often do not push IA concerns early enough in the design process.

Technical Concerns

In one case, IA concerns were addressed in-process for a system-wide infrastructure. These approaches could be applicable to the SoS platform:

- Instead of waiting until all development is complete and ready for end-product certification, the system conducts software IA certification on early releases.
- The system flows IA controls down in initial decomposition. The system achieves a certification board decision within elements before IA involves all stakeholders across elements.
- With external organizations identifying testing requirements and methods, decomposition must proceed to responsibilities at the lowest CI level.

Commonality at an infrastructure level can be independently certified and, if used as-is and in an intended way, should not trigger recertification. Changes to infrastructure or to accepted methods of using the infrastructure might also trigger recertification. More effective assurance practices will become drivers for the SoS and for SoS platforms. But lack of sufficient a priori architecture experience for IA in the systems area will contribute to uncertainty in how much IA is sufficient for a SoS. Within the SoS, boundaries between constituent systems are fuzzy and there is duplicative effort. Often, issues emerge in addressing assurance due to a lack of understanding of SoS architecture and operation, rather than lack of understanding of the specific IA requirements. Addressing emergent behavior may come at the expense of opening a SoS to new IA challenges.

5 Conclusions

This study interviewed 12 experts to characterize the state of the practice of SoS architecture development, with a focus on architectures that include SoS platforms. Our goal was to address the following questions:

1. What processes are used to develop SoS architectures, and how are constituent software elements of the architecture designed?
2. What challenges do SoS programs face in developing architectures; performing test, integration, and assurance; managing runtime configuration and operation; and evolving the SoS? What approaches have been used in successful programs to overcome these challenges?
3. What are the constraints on new approaches to developing, using, and evolving these SoS architectures?
4. What are the important differences between practices used to create commercial SoS architectures and DoD SoS architectures?

The first three goals are addressed by the responses summarized in Section 3, and the tables offer contrasts between categories of organizations—commercial, large-scale command and control SoS for the DoD, and SoS platforms for the DoD. Section 4 offered challenges that the DoD faces in addressing Question 4. This section recommends further research to address the challenges.

5.1 Common Practices in Commercial SoS for DoD Consideration

The interviews uncovered four sets of common practices that we label *proto patterns* [Wiki 2009]. The first three are elements of commercial development that the DoD could adopt. The fourth is an aspect of some DoD developments that could provide assistance across the DoD in evolution to SoS platform adoption.

1. *An architecture evolution proto pattern for evolving a new platform.* This pattern begins by first defining and implementing atomic message types and schemas, with no concept of workflow (i.e., sequences of messages related to a business task or process). Initially, all workflow is organically built into the systems and applications using the platform. Later, workflow orchestration is added to the platform, with the platform providing versioned workflow definitions that include endpoint roles (endpoint cardinality, supported message sets, and other workflows in which the endpoint can participate), workflow sequence definitions, and transaction support. This proto pattern allows an initial version of the platform to be deployed quickly and allows incremental definition of workflows based on actual platform use.
2. *A proto pattern related to the evolution pattern described above.* Workflow execution scalability and availability is achieved by maintaining the workflow state only in the participating endpoints, not in the platform infrastructure. This “stateless platform” approach is a refinement of stateless services in service-oriented architectures.
3. *A three-step approach to address the problem of how to deploy new platform features.* Commercial SoS platform architects also reported a proto pattern for deploying new platform features. This three-step pattern begins by piloting a new feature with selected customers, with special IT operations processes used to carefully monitor usage and quality attributes such as

performance. In a second release, the feature is stabilized with those customers, and IT operations processes are similar to standard production processes. Finally, in a third release, the feature is generally available to all customers in production. (In the organization using this proto pattern, the time from pilot to general availability of a feature was four to eight months.)

4. *Architecture practices evolving from "backing into" SoS architecture.* More developments emphasize quality attributes early in the development, recognizing that they cannot be easily addressed later in the lifecycle. Organizations are using an Architecture Tradeoff and Analysis Method® (ATAM®) evaluation proto pattern to identify sensitivity points, issues in addressing quality attributes, and potential architectural gaps. The evaluation may be coupled with software infrastructure standards such as TSCE-I (Total Ship Computing Environment – Infrastructure), CANES (Consolidated Afloat Network Enterprise Services), SWFTS (Submarine Warfare Federated Tactical System), and the COE to provide an open infrastructure for developer support across the SoS.

5.2 DoD Achieving the Successes of the Commercial World

Many factors exist, or are perceived to exist, that prevent the DoD from duplicating the successes in the commercial world:

- Leadership in the DoD experiences more rapid turnover. Industry has longer leadership tenure.
- The political situation at any particular time—"the surge" one year, sequestration the next—becomes the current hot-button issue and replaces a long-term agenda.
- Suppliers to the DoD cannot be easily replaced due to legal issues and high contract cancellation costs.
- In Title 10 and Title 40, governance restricts the ability to create an "app store" ecosystem. Many developments bypass standard development procedures through Urgent Operational Need (UON) statements. The urgency of a current need can trump a good design.

Existing DoD platform architecture initiatives place priority on enabling quality attributes. These include interoperation among applications and a broad range of requirements for performance and availability across the platform-supported SoS that require dependable and efficient support. Still, the platform developments have brought forth specific challenges:

- *Struggle with interoperability*. Examples include poor data modeling and semantic mismatch on shared data elements.
- *Defining the platform based on warfighter context rather than technical need.* Examples include computing infrastructure for a submarine or a missile cruiser. Both are "real-time" systems, but sensors work at different time scales (speed of sound in water and torpedo velocity vs. radar and missile velocity). Each community thinks that there can't be commonality, but a single scalable platform could serve both.
- *Bridging from system concerns to SoS concerns.* System designers don't know SoS concerns; their focus is on their own products. Systems are tightly aligned with warfighter missions, while the SoS is "someone else's problem."

® Architecture Tradeoff and Analysis Method and ATAM are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

- *Diversity of behaviors in making tradeoffs.* Generally, short-term interests have a higher priority than long-term interests. Programs will invest in local capabilities, system behavior, or performance over SoS concerns.

The DoD does hope to achieve true competition for acquiring not just the infrastructure but also the capability development—"third-party product integration." The apps store concept, as an example for attracting new players, looks attractive, but, as yet, the DoD apps market remains relatively small, with no incentive to enter. Today, many programs advocate openness as a mechanism to encourage competition, but openness is defined differently for different systems. Without qualification, that is, openness with respect to system, SoS, or platform integration, the term does not adequately define a particular quality.

5.3 Recommendations for Further Study

This study identified several areas where additional research is needed.

1. *Selection of features for a SoS platform.* The study identified a number of critical stakeholder concerns, including time to market, cost reduction, ease of adoption, support for future capabilities, and alignment with SoS assurance and certification processes. Feature selection requires consideration of both the problem space, to identify candidate platform features and assess their value, and the solution space, to assess costs to implement and maintain each feature. Systematic approaches to analyze the problem space might combine techniques such as mission thread analysis [Kazman 2012] with domain analysis [Northrop 2012]. Solution space analysis might include economic models and models that consider alignment of the architecture with constraints such as acquisition strategy, organizational structures, and other socio-technical factors. These analyses would be facilitated by catalogs of architecture knowledge, such as pattern handbooks, to provide a repertoire of solutions that exhibit particular functional and quality attribute properties. Finally, a systematic approach, such as economic modeling, is needed to prioritize and select features for inclusion in the platform from a set of candidates.
2. *Agile development methods for platform-based systems of systems.* Approaches for architecture-led incremental development have focused primarily on the software and system level [Bachmann 2012, Bellomo 2013]. Further work is needed to model the more complicated dependencies in a SoS architecture and to develop iteration planning strategies that accommodate the managerial independence of the constituent systems.
3. *Approaches to characterize and document constituent systems* to support their use in systems of systems. Systematic approaches are needed to identify the relevant concerns and collect and present the information to efficiently satisfy those concerns. An approach such as the creation of an ISO 42010-style architecture description viewpoint may be appropriate.

Appendix Interview Outline

1. Demographics
    1.1. Years of professional experience
    1.2. Years of SoS experience
    1.3. Number and names of SoS projects and programs worked on

2. SoS Architecture Development

We are looking here for the architecture design process, not the documentation process (e.g., DoDAF).

For each question, ask interviewee specifically about software, as appropriate.

Try to be as specific as possible.

2.1. In your experience, how are SoS architectures defined? Is software treated separately or differently from the rest of the SoS?

2.2. What is the architecture design process? How do you bridge from SoS-level focus to system-level focus?

2.3. How are architecture design tradeoffs defined or framed? How do you balance SoS context with system context?

2.4. How are architecture design tradeoff decisions made? What factors (technical and non-technical) influence decisions? How do you balance SoS context with system context?

3. SoS Success Patterns and Challenges

Ask interviewee to refer to a specific system, SoS, or program in answering these questions and to be as specific as possible in describing the success or challenge.

We do not want to dig into root-cause analysis—we are looking for the observable events.

When the interviewee identifies a system or program that went well, probe into how he or she defined success—e.g., on budget, on schedule, exceeded requirements, provided a platform that other programs could build upon, or high benefit for cost.

Challenges could include complying with standards, not complying with standards, silos, not-invented-here syndrome, legacy constraints, etc.

3.1. In developing constituent systems for use in a SoS, was there a system or program that went well or was more successful than others? Why? (i.e., what evidence is there?)

3.2. What SoS-related challenges have you seen in the development of constituent systems for use in a SoS?

3.3. In the test, integration, and assurance of constituent systems into a SoS, was there a system or program that went well or was more successful than others? Why? (i.e., what evidence is there?)

3.4. What SoS-related challenges have you seen in the test, integration, and assurance of constituent systems into a SoS?

3.5. In the configuration and management of a SoS, was there a system or program that went well or was more successful than others? Why? (i.e., what evidence is there?)

3.6. What SoS-related challenges have you seen in the configuration and management of a SoS?

For the next two questions, include issues about evolution of a constituent system within the overall SoS.

3.7. In the post-deployment sustainment and evolution of a SoS, was there a system or program that went well or was more successful than others? Why? (i.e., what evidence is there?)

3.8. What SoS-related challenges have you seen in the post-deployment sustainment and evolution of a SoS?

4. Solution Constraints

"Solution" is used in a very general sense here and refers to something that would address one of the *challenges* identified above.

Constraints can be technical, organization, governance, policy, regulatory, incentive, doctrine, or other cause.

4.1. What constraints do you see for a solution that improves the development of constituent systems for use in a SoS?

4.2. What constraints do you see for a solution that improves the test, integration, and assurance of constituent systems for use in a SoS?

4.3. What constraints do you see for a solution that improves the configuration and management of a SoS?

References

URLs are valid as of the publication date of this document.

[Bachmann 2012]
Bachmann, F., Nord, R. L., & Ozkaya, I. "Architectural Tactics to Support Rapid and Agile Stability." *CrossTalk 25,* 3 (May/June 2012): 20–25.

[Bellomo 2013]
Bellomo, S., Nord, R., & Ozkaya, I. "A Study of Enabling Factors for Rapid Fielding: Combined Practices to Balance Speed and Stability," 982–991. *Proceedings of the 35th International Conference on Software Engineering (ICSE 2013)*. San Francisco, CA, May 2013. IEEE, 2013.

[Clark 2008]
Clark, J. O. "System of Systems Engineering and Family of Systems Engineering from a Standards Perspective," 414–419. *Proceedings of the IEEE International Conference on System of Systems Engineering (S0SE 08)*. Monterey, CA, June 2008. IEEE, 2008.

[Cusumano 2010a]
Cusumano, M. "The Evolution of Platform Thinking." *Communications of the ACM 53,* 1 (January 2010): 32–34.

[Cusumano 2010b]
Cusumano, M. "Staying Power: Six Enduring Principles for Managing Strategy & Innovation in an Uncertain World." Presented at the MIT Research and Development Conference. Cambridge, MA, November 2010. http://ilp.mit.edu/images/conferences/2010/RD/Cusumano.pdf

[Dahmann 2011a]
Dahmann, J. & Baldwin, K. "Implications of Systems of Systems on System Design and Engineering," 131–136. *Proceedings of the 2011 6th International Conference on System of Systems Engineering.* Albuquerque, NM, June 2011. IEEE, 2011.

[Dahmann 2011b]
Dahmann, J., Rebovich, G., Lane, J., Lowry, R., & Baldwin, K. "An Implementers' View of Systems Engineering for Systems of Systems," 212–217. *Proceedings of the IEEE Systems Conference (SysCon 2011)*. Albuquerque, NM, April 2011. IEEE, 2011.

[Forsberg 1991]
Forsberg, K. & Mooz, H. "The Relationship of System Engineering to the Project Cycle," 57–65. *Proceedings of the First Annual Symposium of National Council on System Engineering*, Chattanooga, TN, October 1991. Wiley, 1991.

[Frank 2008]
Frank, G., Evens, N., Hubal, R., & Whiteford, B. "Automated, Interactive AARs: A Positive Spin," Paper 8258. *Proceedings of the Interservice/Industry Training, Simulation, and Education*

Conference (I/ITSEC). Orlando, FL, December 2008. RTI International, 2008. http://www.rvht.net/pubs/8258.pdf

[Gorod 2008]
Gorod, A., Sauser, B., & Boardman, J. "System-of-Systems Engineering Management: A Review of Modern History and a Path Forward." *IEEE Systems Journal 2*, 4 (December 2008): 484–499.

[Greenert 2012]
Greenert, J. W. "Payloads over Platforms: Charting a New Course." *U.S. Naval Institute Proceedings Magazine 138*, 7 (July 2012): 16–23.

[INCOSE 2006]
International Council on Systems Engineering. "Definition of a System." *A Consensus of the INCOSE Fellows*. INCOSE, 2006. http://www.incose.org/practice/fellowsconsensus.aspx

[INCOSE 2011]
International Council on Systems Engineering. *INCOSE Systems Engineering Handbook v. 3.2.2* (INCOSE-TP-2003-002-03.2.2). INCOSE, 2011.

[Kazman 2012]
Kazman, R., Gagliardi, M., & Wood, W. "Scaling up Software Architecture Analysis," *Journal of Systems and Software 85*, 7 (July 2012): 1511–1519.

[Klein 2012]
Klein, J., Chastek, G., Cohen, S., Kazman, R., & McGregor, J. "An Early Look at Defining Variability Requirements for System of Systems Platforms," 30–33. *Proceedings of the Second International Workshop on Requirements Engineering for Systems and Systems-of-Systems (RESS 2012)*. Chicago, IL, September 2012. IEEE, 2012.

[Klein 2013]
Klein, J., Cohen, S., & Kazman, R. "Common Software Platforms in System-of-Systems Architectures: The State of the Practice," 135–140. *Proceedings of the 8th International Conference on System of Systems Engineering*. Maui, HI, June 2013. IEEE, 2013.

[Maier 1998]
Maier, M. "Architecting Principles for Systems-of-Systems." *Systems Engineering 1*, 4 (1998): 267–284.

[Maier 2005]
Maier, M. "Research Challenges for Systems-of-Systems," 3149–3154. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC 2005)*. Waikoloa Village, HI, October 2005. IEEE, 2005.

[McConnell 2010]
McConnell, J. H. & Jordan, L. L. *Naval Integrated Fire Control–Counter Air Capability-Based System of Systems Engineering.* Naval Integrated Fire Control – Counter Air (NIFC-CA), 2010. http://www.acq.osd.mil/se/webinars/20100930-Leading-Edge-Input_NIFC-CA.pdf

[Mell 2011]
Mell, P. & Grance, T. *The NIST Definition of Cloud Computing* (Special Publication 800-145). National Institute of Standards and Technology, 2011.

[Northrop 2012]
Northrop, L. M. & Clements, P. C. *A Framework for Software Product Line Practice, Version 5.0*. Software Engineering Institute, Carnegie Mellon University, 2012. http://www.sei.cmu.edu/productlines/frame_report/index.html

[NRC 2010]
National Research Council Committee for Advancing Software Intensive Systems Producibility. *Critical Code: Software Producibility for Defense.* National Academies Press, 2010.

[OPEN 2013]
The Open Group. *The Future Airborne Capability Environment (FACE) Consortium*. https://www.opengroup.us/face (2013).

[OUSD 2008]
Office of the Deputy Under Secretary of Defense for Acquisition and Technology, Systems and Software Engineering. *Systems Engineering Guide for Systems of Systems, Version 1.0.* ODUSD(A&T)SSE, 2008. http://www.acq.osd.mil/se/docs/SE-Guide-for-SoS.pdf

[Raytheon 2006]
Raytheon. *Raytheon Selects RTI Real-Time Middleware for U.S. Navy Destroyer Program*. http://www.rti.com/company/news/raytheon.html (2006).

[Rochet 2006]
Rochet, J.-C. & Tirole, J. “Two-Sided Markets: A Progress Report.” *RAND Journal of Economics 37*, 3 (2006): 645–667.

[U.S. Army 2012]
U.S. Army. *Army Releases Common Operating Environment Release Plan*. http://www.army.mil/article/71710/TSCE-I (2012).

[Wiki 2009]
Portland Pattern Repository Wiki. *Proto Pattern.* Cunningham & Cunningham Inc., 2009. http://c2.com/cgi/wiki?ProtoPattern

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.			
1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE April 2015		3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE State of Practice Report: Essential Technical and Nontechnical Issues Related to Designing SoS Platform Architectures			5. FUNDING NUMBERS FA8721-05-C-0003
6. AUTHOR(S) Sholom Cohen, John Klein			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213			8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2015-TR-007
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFLCMC/PZE/Hanscom Enterprise Acquisition Division 20 Schilling Circle Building 1305 Hanscom AFB, MA 01731-2116			10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES			
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS			12B DISTRIBUTION CODE
13. ABSTRACT (MAXIMUM 200 WORDS) This report presents an analysis of the state of the practice in system-of-systems (SoS) development. SoS architectures, or blueprints for integrating multiple systems based on common software platforms, have been successful in many commercial environments. The report discusses technical issues related to SoS common platform development and adoption in the Department of Defense (DoD) and the nontechnical constraints that must be satisfied. The analysis is based on information captured from 12 interviews of leading SoS developers in the DoD and industry, applying a SoS definition from the literature to identify gaps between the current state and the desired end state. The results of the study show that while commercial and DoD developers follow different approaches, all organizations report nontechnical constraints as more challenging than technical issues. For the DoD, these include leadership changes, shifting political priorities, and difficulty in replacing suppliers. The report recommends further study of SoS planning and agile approaches that better support incremental development; bridging the gap from SoS to system concerns so that system designers understand SoS concerns and can focus on their products in the context of the SoS; and documenting the platform at all software levels, including architecture views and component integration strategies.			
14. SUBJECT TERMS system of systems, SoS, platform, architecture, nontechnical constraints			15. NUMBER OF PAGES 45
16. PRICE CODE			
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL