A Practitioner's Approach using MBSE in Systems of Systems

Richard Deakins
US Army Aviation and Missile Research,
Development and Engineering Center
Colorado Springs, CO

Doug Parsons
US Army Aviation and Missile Research,
Development and Engineering Center
Colorado Springs, CO

ABSTRACT

Recognizing the value of systems engineering (SE) as a key enabler of successful systems acquisition, and the growing importance of systems interdependencies affecting the ability for mission success of highly complex development systems, the Deputy Under Secretary of Defense for Acquisition and Technology developed the [2008] "Systems Engineering Guide for Systems of Systems." While this guide provides excellent insight into the Systems of Systems (SoS) environment, as well as core SoS SE elements, the process to apply them in a DoD acquisition environment is not included. The purpose of this paper is to extend those concepts by defining a system of system acquisition process from receipt of modeling and simulation (M&S) needs through to development of individual requirements for the constituent systems by leveraging the power of Model Based Systems Engineering (MBSE) practices. Because it's not atypical for constituent developers to read the same need statement and interpret what occurs and what is needed on opposite sides of an interface differently, severe issues can result of which are not discovered until the resolution is extremely costly. The use of MBSE and the Systems Modeling Language (SysML) provides formal methods and notations that can remedy SoS misunderstandings prior to development. Further, our proposed process will facilitate collaboration amongst the system constituents and other stakeholders using MBSE throughout the acquisition process will create a shared common understanding and agreement for the efforts required for success of the SoS mission. Included in the discussion of the proposed acquisition process will be conceptual modeling, architecture and design reviews. The nature of DoD missions and the simulations that describe them are becoming more complex with increasing interdependence among the systems involved. This paper intends to provide the practitioner with systems engineer processes that will result in avoidance of the unintended consequences impacting mission success.

ABOUT THE AUTHORS

Richard Deakins is currently the Lead System's Engineer for Missile Defense Agency's Digital Assessment M&S program and lead of the Enterprise Engineering M&S initiative. He has over thirteen years of experience in virtual and constructive simulation development. Mr. Deakins received a B.S. in Computer Science.

Doug Parsons is currently the Lead Software Architect for Missile Defense Agency's Objective Simulation Framework (OSF) program and a key contributor to the Enterprise Engineering M&S initiative. He has over nineteen years of experience in virtual and constructive Engineering simulation development. Mr. Parsons received a B.S. in Mechanical Engineering, a M.S. in Systems Management and a M.S. in Industrial Engineering.

A Practitioner's Approach using MBSE in Systems of Systems

Richard Deakins
US Army Aviation and Missile Research, Development and Engineering Center
Colorado Springs, CO

Doug Parsons
US Army Aviation and Missile Research, Development and Engineering Center
Colorado Springs, CO

INTRODUCTION

In a world of uncertainty for today's Department of Defense (DoD) there exists an inevitability that the services will require more of the systems brought to the battlefield than originally intended. Our expectations will be to deliver decisive victory against a dynamic and ever increasing complex threat, all the while living within tightening budget constraints. To be successful those systems will be employed in such a way to allow the warfighter to 'overcome and adapt' in order to engage and counter whatever the conditions on the ground or sea or air maybe. Increasingly, the employment of a battlefield system will not only operate effectively, but to optimize the success of a mission environment defined in terms of systems-of-systems (SoS). As stated in the DoD SoS Systems Engineering guide [OUSD AT&L, 2008],

"With the adoption of net-centric approach to information management, developers recognize that systems operate in a broader context today than in the past. Most importantly, changing threat situations increase the need for flexibility and adaptability in the way the war fighters configure and apply suites of systems to respond to changing situations. The notion of "systems of systems" is becoming a critical perspective in thinking about systems."

In 2012 the RAND Corporation published a report, "Lessons Learned from the Army's Future Combat Systems," at the request of the Army's Acquisition Executive to provide an after-action analysis of the FCS program [Pernin, et al., 2012]. FCS program officials interviewed for the report stated, "The trend toward networked capabilities will increasingly demand movement away from acquisition of platforms in isolation and toward a more sophisticated consideration of how the Army should integrate systems into existing and future formations." Further, they agreed that more preparatory system engineering is needed for such a large, ambitious program and that SoS engineering should have been much stronger early in the program. And in case the reader believes that programs like FCS are the exception, the Defense Acquisition Guidebook (DAG) states that whether or not a system is formally acknowledged as an SoS, nearly all of our DoD systems function as part of an SoS to deliver capability to the warfighter. Unfortunately, the acquisition of these systems aren't often structured and don't always yield optimized systems-of-systems. There are multiple contributors leading to sub-optimal conditions that include, but are not limited to organizational structure/culture, governance, budgets, and conflicting schedules. While understanding the impact that these contributors have on program and mission success is significant, it is the aim of this paper to focus discussion on how the use of a model based, i.e. Model Based Systems Engineering (MBSE), approach may provide a remedy for some of the technical aspects of this condition.

SYSTEMS OF SYSTEMS PRIMER

The concept of SoS is growing in importance in today's environment. But when is something a system and when is it a system of systems? In helping to distinguish between a system and a system of systems' five key characteristics have been identified; they are operational independence of component systems, managerial independence of component systems, geographic distribution, emergent behavior, and evolutionary development. So where a system is defined as a functionally, physically, and/or behaviorally related group of regularly interacting or interdependent elements; that group of elements forming a unified whole [DoD, Joint Publication 1-]. A SoS is defined as a set or arrangement of interdependent systems that are related or connected to provide a given capability. As stated in the Defense Acquisition Guidebook (DAG), the loss of any part of the system will degrade the performance or capabilities of the whole.

SoS's in their various forms create a unique environment to practice the art of systems engineering (SE). Aspects of SE when applied to a system do not necessarily hold explicitly true applied to a SoS. When considering an acknowledged SoS, in the area of management and oversight, stakeholder involvement is ambiguous vs. explicit as well as governance becomes partial vs. complete. In the area of operational environment, the operational focus is lacking or not fully aligned vs. designed and developed to meet the objective. In the area of implementation, the complexity of contending with multiple systems lifecycles across their acquisition vs. a single acquisition as well as testing across multiple system lifecycles vs. a single lifecycle increases complexity. Finally in the area of engineering and design boundaries and interfaces becomes about the focus enabling the flow of data, control and functionality across the SoS vs. identifying the interface for a single system. Additionally, when considering performance, it is about achieving the performance to meet capability needs while balancing and not impacting the systems performance vs. focused on a single system's performance [Dahmann, et al., 2008]. SoS SE's must be able to function in a more ambiguous, diverse and complex environment.

A SoS when successfully implemented will provide a more varied set of capabilities then that of an individual system.

SYSTEMS OF SYSTEMS CHALLENGES

As challenging as it is working a single system and the associated systems engineering to go with it, working in systems of systems and the associated systems engineering to go with it scale the problems and are that much more challenging. In understanding what Systems of Systems is as well as understanding the unique aspects of systems engineering in SoS as compared to systems, one can begin to see the challenges associated with Systems of Systems. Challenges in SoS's like systems spans from management to technical. Dr. Judith Dahmann describes these challenges as 'pain points' and groups them into seven categories that affect the SoS throughout the development process. These pain points and associated questions are shown in Table 1 [Dahmann, 2012].

Table 1. Systems of Systems Pain Points

Pain Points	Question
Lack of SoS Authorities & Funding	What are effective collaboration patterns in systems of systems?
Leadership	What are the roles and characteristics of effective SoS leadership?
Constituent Systems	What are effective approaches to integrating constituent systems into a SoS?
Capabilities & Requirements	How can SE address SoS capabilities and requirements?
Autonomy, Interdependencies & Emergence	How can SE provide methods and tools for addressing the complexities of SoS interdependencies and emergent behaviors?
Testing, Validation & Learning	How can SE approach the challenges of SoS testing including incremental validation and continuous learning in SoS?
SoS Principles	What are the key SoS thinking principles, skills and supporting examples?

Lack of SoS Authorities, Funding, and Leadership speaks to the set of challenges associated with SoS governance and management and any cross cutting SoS activities. Poorly defined roles and responsibilities of the SoS and the individual systems that make it up are usually ambiguous and not well understood with respect to each other [Baldwin, 2012]. MOAs or other type documents are lacking. Funding and cost management are usually not performed across the SoS but rather at the level of the individual programs responsible for the systems that make up the SoS. There is no sense of incentives or rewards for the individual systems in the evolving SoS [Sledge, 2010].

Constituent Systems speaks to the set of challenges associated with the heritage of the individual systems within the SoS. An individual system is designed and developed by definition for a specific purpose and context. The technical issues to overcome in order to align not just one individual system but all of the identified individual systems associated with a SoS to an SoS architecture while staying true to their original purpose is complex and fraught with interoperability issues.

Capabilities and Requirements speak to the set of challenges associated with the high level capability needs and their definition in order to support requirements decomposition and allocation to individual systems which may or may not be willing to address them. Having insufficient knowledge or lack of understanding on how the SoS would actually be used (eg CONOPs) vs. just the individual systems [Sledge, 2010].

Autonomy, Interdependencies and Emergence speak to the set of challenges associated with the complexity of individual systems autonomy, the SoS's interdependencies and the unpredictable nature of emergent behavior by the SoS. It is often difficult to understand and determine which individual systems are core to the SoS, which can be added over time, and which drive emergent behavior. What are the impacts from individual systems within the SoS evolving on other individual systems within the SoS and the SoS itself?

Testing, Validation and Learning speaks to the set of challenges associated with the individual systems upgrading and evolving based on their primary needs and users independent of the SoS as well as when it is ideal for the particular individual system making up the SoS and being able to fully test the SoS in a representative environment. Trying to coordinate and test the SoS and validate the results with constantly changing individual systems is an integration nightmare.

SoS principles speak to the set of challenges associated with the general lack of understanding SoS's as well as any successful examples. Applying standard SE processes and practices without understanding the unique nature of the SoS environment increases the likelihood of issues and the risk of failure.

SOS ENGINEERING PROCESS

One of the advantages in using MBSE in defining a SoS is the development of a common understanding of SoS mission and inter-relationships and intended capabilities required of the individual constituent systems. Still, an MBSE methodology and toolset alone is insufficient in assuring that this advantage is captured. Figure 1 is a description of a process that facilitates collaboration, management, development of architectural diagrams, and generation of the appropriate system level requirements.

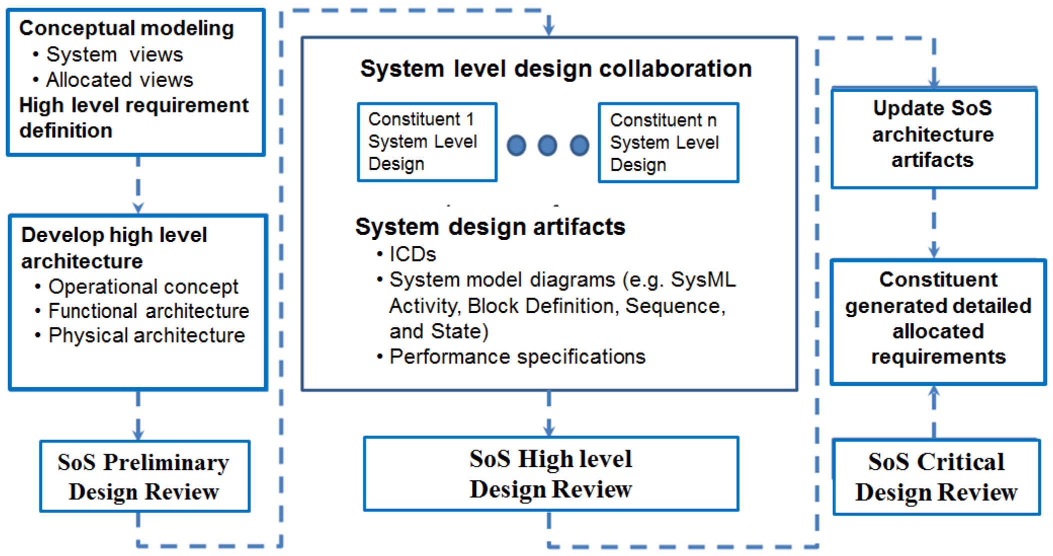
Conceptual modeling
• System views
• Allocated views
High level requirement definition
Develop high level architecture
• Operational concept
• Functional architecture
• Physical architecture
SoS Preliminary Design Review
System level design collaboration
Constituent 1 System Level Design
Constituent n System Level Design
System design artifacts
• ICDs
• System model diagrams (e.g. SysML Activity, Block Definition, Sequence, and State)
• Performance specifications
SoS High level Design Review
Update SoS architecture artifacts
Constituent generated detailed allocated requirements
SoS Critical Design Review

Figure 1. SoS Engineering Process

Define the area of interest and interactions. The first step in the process is developing a conceptual understanding of the mission space defining the intended capabilities for the SoS use case. The question being addressed at this point is, "How do the constituent systems fit into the SoS mission?" Conceptual modeling builds a bridge between the real-world systems and the simulation environment. The activities include gathering information, identification of gaps and overlaps, resolution of conflicts, and setting scope, boundaries and quality attributes. The result of this effort is a set of views describing the entities, behaviors, and interactions of the actual systems abstracted appropriately for modeling of the constituent systems. Development of high level requirements of the desired capability occurs in concert with conceptual modeling and defines the *"Should-Be"* state of the SoS.

Develop high level architecture. The second phase of the process describes how the SoS is intended to be used and the initial development of the structure to support a given use case. Operational artifacts may include a Concept of Operations and use case scenarios that identify actors and their interactions. The functional and physical architectural artifacts describe the SoS in an *"As-Is"* state.

SoS Pain Point Addressed - SoS Principles

As a result of developing a high level architecture the very nature of the SoS environment, as well as its associated challenges, will be much better understood.

System level design collaboration. The authors recommend that development of the high level architecture be led and executed by a SoS defined engineering team supported by constituent system models and frameworks subject matter experts (SME) as needed. During the system level design phase this team becomes more of a facilitator. The goal during this phase is to collaboratively design the *"To-Be"* architecture based on artifacts from the previous phases. The purpose of the SoS engineering team is not to design the changes required of the constituent system models and frameworks to support SoS capabilities. As such, these SMEs are best suited to work together to evolve a SoS to achieve the desired future state. The artifacts resulting from this phase include changes and extensions to the high level architectural diagrams, plus the appropriate Interface Control Documents (ICD) and performance specifications.

SoS Pain Point Addressed – Lack of SoS Authorities & Funding

Part of this pain point concerns itself with effective collaboration. The system level design process specifically executes methods to facilitate collaboration among the constituent systems. In addition, an understanding of the SoS as a whole rather than a collection of pieces supports cost management at that level.

Update SoS architectural artifacts and generate detailed, allocated requirements. Following the system level design phase there exists the possibility of discovery that require modification of the original high level architecture diagrams to maintain consistency. It is in this phase that uniformity is assured prior to the generation of constituent system requirements. Ultimately, the objective of the entire SoS design process is to develop a set of unambiguous and comprehensive set of detailed requirements that are allocated to the constituent systems. In addition, these requirements will be accompanied by a set of architectural artifacts that will guide their design efforts.

SoS Pain Point Addressed – Capabilities & Requirements

The main objective of this process is specifically aimed at aligning top level SoS capability needs with requirements supported by the constituent systems.

Design reviews. In order to provide opportunity for management oversight and decision authority the authors built in three design reviews. The Preliminary Design Review (PDR) allows for a review of high level architecture diagrams and a decision to proceed to the collaboration phase. The High level Design Review (HDR) performs a review of the constituent level diagrams and makes a readiness decision to generate the associated requirements. The Critical Design Review (CDR) creates a final decision point prior to proceeding to development of constituent capabilities supporting the SoS mission. These design reviews can be tailored to suit any type of technical review best suited to support the need of the SoS program decision-making process.

SoS Pain Point Addressed – Leadership

The inclusion of technical design reviews offers the opportunity for SoS leadership to learn of the status of the design effort and put in place governance and other management policies that support decision-making.

MBSE PRIMER

In order to provide a better understanding of Model Based Systems Engineering and how it supports Systems of Systems a brief summary of terms, definitions and concepts is provided here.

As defined by INCOSE, Model Based Systems Engineering is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases." [INCOSE, 2007] The goal of the MBSE process is the creation of a well-reasoned and consistent system model. The same could be said of a conventional systems engineering process. The distinction is that traditional systems engineering tends to be centered on a static, document-based approach in the form of design documents and text specifications. Certainly, systems engineering to date has proven to be thorough, but has some fundamental limitations. In a *Practical Guide to SysML* Sanford Friedenthal et al. indicate that the completeness, consistency, and relationships between requirements, design, engineering analysis, and test information are difficult to assess since this information is spread across several document. Further, this makes it difficult to understand a particular aspect of the system and to perform the necessary traceability and change impact assessments. Instead, MBSE is focused on a shared system model with multiple views. MBSE can leverage the Object Management Group's (OMG) Systems Modeling Language (SysML) as a graphical modeling language for representing system's structure, behavior, requirements and parametrics. Software engineering practitioners will note that SysML is an extension of the Unified Modeling Language (UML). As a standardized language with robust semantic foundation, commercial MBSE tools are available to create and maintain a SysML-defined system model. MBSE views may include the following SysML diagrams.

Activity diagrams – Specify transformation of inputs to outputs through a controlled sequence of SoS actions described by constituent system models.

Sequence diagrams – Provide representations of message based behavior representing the flow of control and interactions between the constituent system models.

Use case diagrams – Provide a description of basic functionality in terms of goals of the constituent system models supporting the SoS use case.

State machine diagrams – Represents event-based behaviors of the constituent system models within the SoS (i.e. state transitions and changes)

Block definition diagrams – Describes the structure of the SoS in terms of relationships among the constituent system models properties, operations, constraints, allocations, etc.

Internal block diagrams – Describes the internal structure of the SoS in terms of the properties and connectors between constituent system models.

MBSE APPLIED TO SYSTEMS OF SYSTEMS SOLUTIONS

MBSE has primarily been applied within the context of understanding the internal operations of a given system and how it interacts with adjacent systems, many and perhaps most of its capabilities can be applied to a SoS environment. The following benefits are purported for MBSE of a single system and should be equally or even more beneficial when applied to a SoS. MBSE enhances the ability to capture, analyze, share, and manage the information associated with the complete specification of a product, resulting in the following benefits [INCOSE, 2006].

- *Improved communications among the development stakeholders (e.g. the customer, program management, systems engineers, hardware and software developers, testers, and specialty engineering disciplines).* Because SoS can cross organizational and/or program boundaries, each with their own methods, techniques and formats,

- misunderstood intentions and expectations can occur more frequently. Often these miscommunications are discovered at a time when schedule and resource impacts lead to tensions or hostilities between organizations.
- *Increased ability to manage system complexity by enabling a system model to be viewed from multiple perspectives, and to analyze the impact of changes.* In most cases the individual systems within a SoS are already extremely complex. Requiring that multiple complex systems interact and interoperate together increases the level of complexity. Complexity can lead to the unintended consequences usually not discovered until integration and test requiring costly rework.
- *Improved product quality by providing an unambiguous and precise model of the system that can be evaluated for consistency, correctness, and completeness.* Ambiguity in requirements can be especially damaging in a SoS environment. Not surprisingly, three teams representing three individual systems that must interoperate together to achieve SoS mission success can look at the same requirements description and walk away with three separate notions of what is expected of them based on their own perspective.
- *Enhanced knowledge capture and reuse of the information by capturing information in more standardized ways and leveraging built-in abstraction mechanisms inherent in model-driven approaches. This in turn can result in reduced cycle time and lower maintenance costs to modify the design.* Reuse in SoS, particularly for simulation use cases, can be especially prevalent when the difference between training or analytical events, for example, do not require extensive redevelopment of the SoS. MBSE methods support formats and tools that lend themselves well to modification of a portion of a SoS and understanding the cascading effects across the SoS.
- *Improved ability to teach and learn systems engineering fundamentals by providing a clear and unambiguous representation of the concepts.* Systems Engineering of complex systems is already a challenging task and SoS increases the complexity. Model-based techniques provides the engineers trying to describe those systems with a toolkit that allows the SoS to be visualized through models (versus lengthy textual descriptions) and thereby making the message more clear and less ambiguous.

Leveraging MBSE in the SoS Design Effort

The objective of the SoS design effort is the development of a set of mutually developed requirements and architectural artifacts that the constituent system developer can proceed unambiguously into their own respective designs. There are a variety of MBSE methodologies that can be employed to complete the design effort. Figure 2 is a tailored step taken from the Object Oriented Systems Engineering Method (OOSEM), specifically ‘Specify and Design System’. The most significant outcomes of this process are the development of a logical and physical architecture. Together, these artifacts describe how the constituent systems interact and behave to achieve SoS requirements.

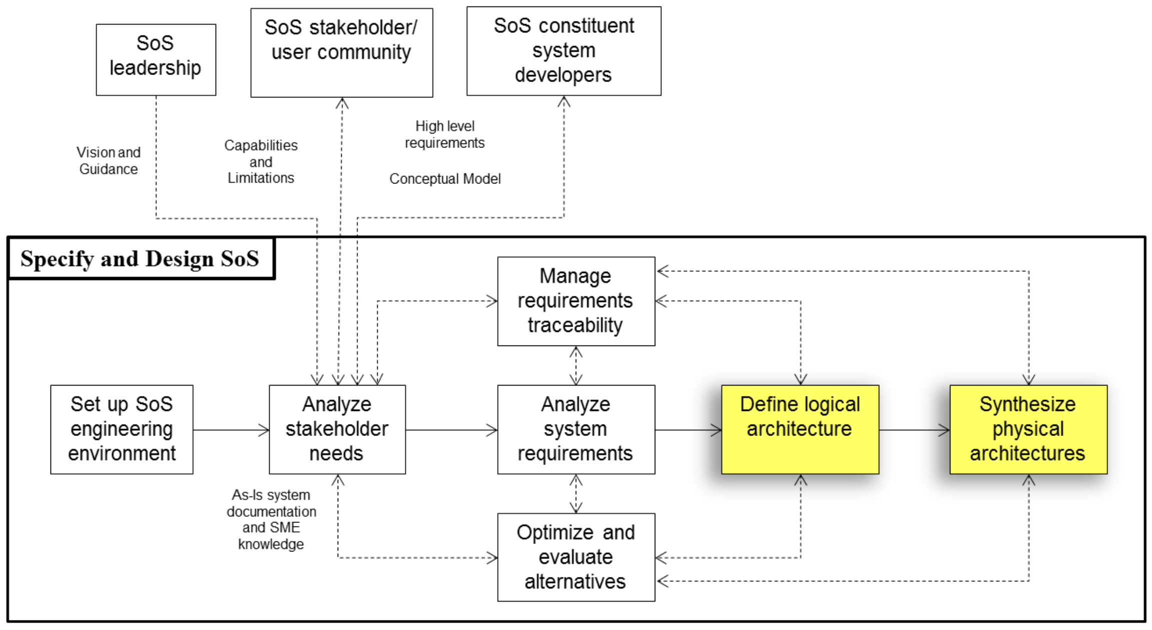
SoS leadership
SoS stakeholder/ user community
SoS constituent system developers
Vision and Guidance
Capabilities and Limitations
High level requirements
Conceptual Model
Specify and Design SoS
Manage requirements traceability
Set up SoS engineering environment
Analyze stakeholder needs
Analyze system requirements
Define logical architecture
Synthesize physical architectures
As-Is system documentation and SME knowledge
Optimize and evaluate alternatives

Figure 2. SoS Design Process

Logical architecture. The purpose for developing a logical architecture is to decompose the SoS into logical components that define their interactions and interconnections. At this point the components are abstractions of SoS functions without imposing implementation considerations and constraints. Doing so allows the designer to work with abstractions that allow greater flexibility in identifying possible solutions. SysML activity diagrams can be developed from SoS scenarios. These diagrams show how the logical components interact to perform the various activities. In addition, SysML Internal Block Diagrams can be developed to describe interconnections between logical components.

SoS Pain Point Addressed – Constituent Systems

The process of developing a logical architecture forces the conversation to first define what is needed of SoS capabilities, which can be in terms of constituent systems interoperating together rather than a collection of individual systems.

Physical architecture. In this phase the SoS designers leverage the logical architecture to allocate physical components (hardware, software, data, etc.). Ultimately, the physical architecture describes the SoS by the physical components and relationships in relation to the respective constituent systems. Physical node architecture is created to allocate logical components to physical components. Activity diagrams can be developed to illustrate interaction of components and how each activity is executed by the physical components. Again, Internal Block Diagrams can be used to show how physical components are interconnected. As part of this phase hardware and software architectures can be developed, however, these are best reserved for the designers of the individual constituent systems to perform. Of significant importance for design is the development of the data architecture, particularly for a SoS. The data architecture is a view of the physical architecture that represents the persistent data, how the data is used, and where the data is stored. Interface Control Documents (ICD) become critical artifacts resulting from this phase, as well as the supportive diagrams.

SoS Pain Point Addressed – Autonomy, Interdependencies & Emergence

The purpose of developing a physical architecture is to define the necessary components and their interrelationships by understanding the interdependencies and behaviors required to produce a desired SoS effect. The use of SysML toolsets to define the SoS architectures (logical, as well as physical) helps contain the complexity.

SoS Pain Point Addressed – Testing, Validation & Learning

The process of developing a physical, as well as a logical architecture, allows a better understanding of the interrelationships among the constituent system functions and possible interactive effects from a SoS perspective. This level of architectural rigor will foster more robust testing and validation.

This paper should have demonstrated that, while not a panacea for resolving all aspects of SoS pain points, there exist techniques through collaborative engineering processes and robust model based systems engineering methods for a practitioner to mitigate the challenges.

MBSE BENEFITS TO THE SIMULATION DOMAIN

The earlier FCS example may leave some readers with the impression that MBSE exists primarily in the operational SoS development domain. Quite to the contrary, the very nature of simulations supporting operational SoS environment for a variety of use cases (e.g. training, analysis, and wargaming) makes it more complicated and more in need of model-based approaches. Over the years systems engineering processes supporting operational, as well as simulation, system development has proven to be rigorous and quite valuable. In a 2010 task group report published by the National Defense Industrial Association (NDIA) [NDIA, 2006] cites SoS as part of the top 5 systems engineering issues within the defense industry stating, "current software development and management practices have not kept pace with emerging needs in areas such as complex systems, Systems of Systems, system assurance, software verification, and COTS/NDI integration." Because simulations are abstractions of the operational environment representing differing use cases there is an added level of complexity in essentially accounting for both domains. Representing a potentially mixed live, virtual, and constructive environment makes the development of the associated model-based SoS architecture even more essential. Published in 2008 the Live

Virtual Constructive Architecture Roadmap (LVCAR) [Henninger, et al., 2008] provided a set of guiding principles for implementation and execution of the LVC way forward. Model-based solutions can be applied to each of the report's fundamental precepts (shown below) from a SoS perspective.

Fundamental Precept #1: Do No Harm. The report recommends that the DoD should not take any immediate action to discontinue any of the existing simulation architectures. Any attempt by the DoD to mandate a convergence solution on an unwilling user base is certain to meet strong resistance and likely to fail. MBSE can provide a means to understand the impacts made to any one system across the SoS. Understanding is the first step to acceptance.

Fundamental Precept #2: Interoperability is not free. The report points out that the DoD must make the necessary investments to enable implementation of the activities described in the LVC Roadmap. MBSE can provide a basis for understanding the nature of those investments and their subsequent effects.

Fundamental Precept #3: Start with Small Steps. The report recommends that the DoD should take immediate action to improve interoperability among existing simulation architectures. The vast range of technical problems currently associated with the development and execution of mixed-architecture LVC environments is well recognized. Such problems increase the technical risk associated with the use of these mixed-architecture environments, and require considerable resources to address. MBSE allows for incremental changes from an "as-is" architecture containing the legacy models, simulations, and frameworks to a "to-be" architecture.

Fundamental Precept #4: Provide Central Management. The report offers that the DoD must establish a centralized management structure that can perform Department-wide oversight of M&S resources and activities across developer and user organizations. MBSE provides a methodology and associated tools that will allow appropriate centralized decision-making across the SoS.

SUMMARY

The increasing trend for military operations is the orchestrated execution of multiple complex systems, all of which are interdependent for mission success. The world of simulation is equal to and usually much greater from a SoS perspective. A simulation SoS event has all of the complex interdependent models describing the participating constituent weapon systems, but also includes considerations for environmental models, communication models, and architectural frameworks. The Department of Defense has lessons learned of the challenges and appreciates the extreme importance for making improvements to better understand the complexity of SoS's. As one point of light example the Office of the Deputy Assistance Secretary of Defense (ODASD) for Systems Engineering facilitates web-based collaborative exchanges from members of the defense acquisition community to include government, industry, and academia. The authors of this paper encourage readers to participate in this community (http://www.acq.osd.mil/se/outreach/sosecollab.html). Even the Defense Advanced Research Projects Agency (DARPA) recognizes the complexity in overcoming SoS challenges and has recently solicited innovative proposals to demonstrate that a SoS approach can provide increased military effectiveness, cost leverage, and adaptability. This paper shows that the initial step in the evolution for superior SoS's need not be "DARPA hard". The use of model based techniques applied to SoS's can significantly increase the community's understanding of the complex interactions, aid in the development and employment of constituent systems (weapon platforms and simulation models/frameworks), and mitigate associated risks.

REFERENCES

Baldwin, K. (2012). System of Systems (SoS) Systems Engineering in Acquisition Program Planning. Presentation for the NDIA Systems Engineering Conference, 24 October 2012. Retrieved from http://www.acq.osd.mil/se/briefs/14722-2012_10_24-NDIA-SEC-Baldwin-SoS-Acq-Prog-Planning.pdf.

Dahmann, J., Lane, J., Rebovich, G., & Baldwin, K. (2008), Retrieved from http://www.acq.osd.mil/se/docs/2008-04-04_CSER-Paper_Dahmann-etal-SoS.pdf.

Dahmann, J. (2012). SoS Pain Points & Implications for MBSE. Presentation at the INCOSE International Workshop, 26-29 January 2013 . Retrieved from http://www.ndia.org/Divisions/Divisions/SystemsEngineering/Documents/NDIA-SE-MS-SoS_2013-08-20_Dahmann.pdf.

Defense Acquisition University (DAU). Systems of Systems. *Defense Acquisition Guidebook*, Chapter 4, Section 4.2.1.2. Retrieved from https://acc.dau.mil/CommunityBrowser.aspx?id=638308&lang=en-US.

Department of Defense (2010). Joint Publication 1-02, *Department of Defense Dictionary of Military and Associated Terms*. Retrieved from http://www.dtic.mil/doctrine/new_pubs/jp1_02.pdf.

Friedenthal, S., Moore, A., Steiner, R. (2012). *A Practical Guide to SysML*. Waltham, MA: Morgan Kaufmann OMG Press.

Hardy, D. (2006). Model Based Systems Engineering and How It Aids DoD Acquisition & Systems Engineering (Presentation). Office of the Under Secretary of Defense (A&T). Retrieved from http://www.dtic.mil/ndia/2006systems/Tuesday/hard.pdf.

Henninger, A., Cutts, D., Loper, M., Lutz, R., Richbourg, R., Saunders, R., Swenson, S. (2008). Live Virtual Constructive Architecture Roadmap (LVCAR) Final Report. *Institute for Defense Analyses*. Retrieved from http://www.msco.mil/files/MSCO%20Online%20Library/LVCAR%20-%201%20of%205%20-%20Final%20Report%20-%2020090814.pdf.

International Council on Systems Engineering (INCOSE) (2007). Systems Engineering Vision 2020. INCOSE-TP-2004-02. Retrieved from https://www.incose.org/ProductsPubs/pdf/SEVision2020_20071003_v2_03.pdf.

National Defense Industrial Association (NDIA), (2006). Top Software Engineering Issues within the Department of Defense and Defense Industry version 5a. *NDIA Systems Engineering Division Task Group Report.* Retrieved from http://www.ndia.org/Divisions/Divisions/SystemsEngineering/Documents/Studies/NDIA%20Top%20SW%20Issues%202010%20Report%20v5a%20final.pdf.

Office of the Under Secretary of Defense for Acquisition, Technology and Logistics (OUSD AT&L), (2008). Systems Engineering Guide for Systems of Systems version 1.0. Retrieved from http://www.acq.osd.mil/se/docs/SE-Guide-for-SoS.pdf.

Pernin, C., Axelband, E., Drezner, J., Dille, B., Gordon IV, J., Held, B., McMahon, S., Perry, W., Rizzi, C., Shah, A., Wilson, P., Sollinger, J. (2012). Lessons from the Army's Future Combat Systems Program. *Rand Corporation*. Retrieved from http://www.rand.org/content/dam/rand/pubs/monographs/2012/RAND_MG1206.sum.pdf.

Sledge, C. (2010). Reports from the Field on System of Systems Interoperability Challenges and Promising Approaches. Software Engineering Institutue Technical Report CMU/SEI-2010-013. Carnegie Mellon University.