2016-06

The system of systems architecture feasibility assessment model

Gillespie, Stephen E.

Monterey, California: Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

THE SYSTEM OF SYSTEMS ARCHITECTURE FEASIBILITY ASSESSMENT MODEL

by

Stephen E. Gillespie

June 2016

Dissertation Supervisor Eugene Paulo

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

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 instruction, 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** June 2016	**3. REPORT TYPE AND DATES COVERED** Ph.D. Dissertation	
4. TITLE AND SUBTITLE THE SYSTEM OF SYSTEMS ARCHITECTURE FEASIBILITY ASSESSMENT MODEL		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Stephen E. Gillespie			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number N/A.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

This research presents the system of systems (SoS) tradespace definition methodology (SoS-TDM) and SoS architecture feasibility assessment model (SoS-AFAM). Together, these extend current model-based systems engineering (MBSE) and SoS engineering (SoSE) methodologies. In particular, they extend the methods of tradespace exploration to considerations of multiple perspectives of an SoS—the physical, process, and organization. In considering multiple perspectives of an SoS, one better defines the SoS and is more likely to correctly represent its performance in an analysis model. The SoS-TDM defines an SoS tradespace by progressively winnowing the design space with increasingly strict definitions of feasibility and then exhaustively analyzing the remaining points. The SoS-AFAM defines and assesses SoS architecture feasibility through a variety of tests that consider the aforementioned aspects of an SoS. Together, these methods may be integrated with existing MBSE and SoSE methodologies and extend their utility.

14. SUBJECT TERMS Model-based systems engineering (MBSE), systems of systems (SoS), systems architecting, systems analysis,			**15. NUMBER OF PAGES** 275
			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** UU

NSN 7540–01-280-5500

Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

THE SYSTEM OF SYSTEMS ARCHITECTURE FEASIBILITY ASSESSMENT MODEL

Stephen E. Gillespie
Captain, United States Army
B.A., Boston University, 2006
M.A., Boston University, 2006

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2016

Approved by:

Eugene Paulo
Associate Professor
Systems Engineering
Dissertation Supervisor

Ronald Giachetti
Professor
Systems Engineering

Rudolph Darken
Professor
Computer Science

Alejandro Hernandez
Associate Professor
Systems Engineering

Paul Beery
Research Associate
Systems Engineering

Approved by: Ronald Giachetti, Chair, Department of Systems Engineering

Approved by: Douglas Moses, Vice Provost for Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This research presents the system of systems (SoS) tradespace definition methodology (SoS-TDM) and SoS architecture feasibility assessment model (SoS-AFAM). Together, these extend current model-based systems engineering (MBSE) and SoS engineering (SoSE) methodologies. In particular, they extend the methods of tradespace exploration to considerations of multiple perspectives of an SoS—the physical, process, and organization. In considering multiple perspectives of an SoS, one better defines the SoS and is more likely to correctly represent its performance in an analysis model. The SoS-TDM defines an SoS tradespace by progressively winnowing the design space with increasingly strict definitions of feasibility and then exhaustively analyzing the remaining points. The SoS-AFAM defines and assesses SoS architecture feasibility through a variety of tests that consider the aforementioned aspects of an SoS. Together, these methods may be integrated with existing MBSE and SoSE methodologies and extend their utility.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ANOVA	Analysis of Variance
DOD	Department of Defense
DODAF	Department of Defense Architecture Framework
DOD CIO	Department of Defense Chief Information Officer
DOE	Design of Experiments
FCS	Future Combat System
FFBD	Function Flow Block Diagram
ICAM	Integrated Computer Aided Manufacturing
IDEF0	ICAM Definition for Functional Modeling 0
IDF	Indirect Fire
INCOSE	International Council on Systems Engineering
LCIM	Levels of Conceptual Interoperability Matrix
MEASA	Method for Employing Architecture in Systems Analysis
MBSE	Model-Based Systems Engineering
MOE	Measure of Effectiveness
MOP	Measure of Performance
OMG	Object Management Group
PCD	Percent Collateral Damage
PTD	Percent Targets Destroyed
ROE	Rules of Engagement
SE	Systems Engineering
SoS	System of Systems
SoS-AFAM	System of Systems Architecture Feasibility Analysis Model
SOSAT	System of Systems Analysis Tool (Sandia National Laboratories)
SoSE	System of Systems Engineering
SoS-TDM	System of Systems Tradespace definition Methodology
SoSTEM	System of Systems Tradespace Exploration Methodology
SysML	Systems Modeling Language
TSE	Tradespace Exploration
UML	Unified Modeling Language

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF MATHEMATICAL NOTATION

Note: The majority of this notation is rigorously defined in Section II.B.2. Notation not defined in that section is defined as it is used in the body of the text.

$\boldsymbol{D_j}$	The domain of the j^{th} parameter
D_j^{min}	The lower allowable bound for the j^{th} parameter
D_j^{max}	The upper allowable bound for the j^{th} parameter
$\boldsymbol{D}$	Design Space, the set of all possible design points
$\boldsymbol{D^A}$	The set of acceptable design points
$\boldsymbol{D^{AP}}$	The set of Pareto optimal points from $\boldsymbol{D^A}$
$\boldsymbol{D^F}$	The set of feasible design points
$\boldsymbol{D^{Phys}}$	The set of points described by the physical parameters
$\boldsymbol{D^{Phys\text{-}F}}$	The set of $\boldsymbol{D^{Phys}}$ that are feasible
$\boldsymbol{D^{Org}}$	The set of points described by the organizational parameters
$\boldsymbol{D^{Org\text{-}F}}$	The set of $\boldsymbol{D^{Org}}$ that are feasible
$\boldsymbol{D^{Proc}}$	The set of points described by the process parameters
$\boldsymbol{D^{Proc\text{-}F}}$	The set of $\boldsymbol{D^{Proc}}$ that are feasible
$\boldsymbol{d_i}$	The i^{th} design point
d_{ij}	The j^{th} parameter of the i^{th} design point
δ_{ai}	The a^{th} attribute of the i^{th} design point, i.e., $f_a(d_i)$
$\boldsymbol{\delta_i}$	The set of all attributes of the i^{th} design point
δ_{a*}^{min}	The lower allowable bound for the a^{th} attribute
δ_{a*}^{max}	The upper allowable bound for the a^{th} attribute
$\boldsymbol{E}$	The set of all possible environmental points
$\boldsymbol{e_m}$	The m^{th} environment
e_{mj}	The j^{th} parameter of the m^{th} environment
$f_a: \boldsymbol{D} \rightarrow \boldsymbol{R_a}$	The a^{th} attribute function
$\boldsymbol{f}$	The set of all attribute functions
$\boldsymbol{R_a}$	The range of the a^{th} attribute
$u_a: \boldsymbol{R_a} \rightarrow [0, 1]$	The utility function for the a^{th} attribute.
$\boldsymbol{u}$	The set of all utility functions
μ_a	The minimum allowable utility for the a^{th} attribute
w_a	The weight of the a^{th} attribute

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

A. BACKGROUND AND CHALLENGE

This dissertation introduces a methodology and a means by which to define a system of systems (SoS) tradespace. A SoS tradespace, to be defined completely, must address the physical, process, and organizational aspects of the SoS architecture. Including these perspectives extends the state-of-the-art for system tradespace development. This extension is accomplished through the contributions of this dissertation, the SoS Tradespace Definition Methodology (SoS-TDM) and SoS Architecture Feasibility Assessment Model (SoS-AFAM).

SoS are a unique class of systems defined as systems composed of operationally and managerially independent constituent systems whose interactions produce a desired emergent behavior (Maier 1998). SoS have been found to meet many organizational needs, particularly those of the Department of Defense (DOD) (DOD 2008); however, the design and development of SoS has proven difficult (Pernin et al. 2012).

The challenge of designing an SoS has several distinctions from that of designing a monolithic system. A significant one is how an SoS ’s architecture must be defined. Desired SoS behaviors and capabilities are emergent properties that arise through the interactions of the constituent systems; accordingly, these interactions form the architecture of the SoS (Maier 1998). These interactions are founded upon the physical—the composition of included systems and their information interfaces; however, as the constituent systems are decision making entities, their interactions are governed by the ways in which these processes relate. Two of these relations may be defined by processes and organizations.

An, or perhaps *the*, important aspect of design is how to choose among potential designs. There are three methods of design decision-making: heuristics, normative, and exploratory. Heuristics are natural language guidelines based upon experience. While useful for quickly reducing ambiguity and contending with complexity, they are limited in that they make no utility of analytic means for a specific problem. Normative decision-

making is the typical analysis seen in most systems engineering or decision science texts. It relies heavily upon pre-established metrics and values. This allows for dispassionate analysis, but presupposes that the metrics and values are inherent and correct (Giachetti and Whitcomb 2016). This, too, has proven lacking in many scenarios. Exploratory design decision-making augments these methods by combining aspects of both.

Exploratory design decision-making is closely coupled iteration of synthesis and analysis—problem framing, solution development, and value analysis done nearly simultaneously. In some cases, this may be done physically through prototypes and similar means as popularized by "design thinking" (IDEO 2016). More analytically, it may be done in a virtual environment in which one couples system designs (architectures) with their attributes (analysis); this is called tradespace exploration (TSE). The methods to do this rigorously in the context of model-based systems engineering (MBSE) are areas of current research (Beery 2016).

A tradespace is the set of all possible designs that can be developed, the attributes of these designs (e.g., cost or performance), and the set of bounds that define what is and is not allowable. This may be described mathematically. Each design point may be defined as a vector where each entry is a parameter that describes it. Associated with each design point are a number of attributes; these attributes are matched to design points via attribute functions. Each parameter and attribute are defined on some domain; the set of acceptable bounds vary these domains. The problem in defining a tradespace, therefore, is in defining the design space and the attribute functions.

To define the design space for an SoS , one must include parameters that describe the SoS from a physical, process, and organizational perspective. This is necessary, as these three vantages are required for a complete SoS architecture. Not only does this correctly define an SoS architecture, but also this is useful for SoS design analysis. These parameters inform SoS models and simulations, such as agent based models (ABM), which require input to define how systems (agents) interact in the model's context. In itself, defining an SoS in this manner is not difficult, though it has not been done for SoS in a TSE environment.

The real challenge of any tradespace definition problem is in defining the attribute functions. This is because design spaces are large (i.e., there are a significant number of design points in them) and the time to assess all of these points is not. Even with very fast computers, the size of the design space may quickly become too large for exhaustive analysis. In many cases, it is possible to approximate these attribute functions; however, due to the complex nature of the interactions in an SoS , this is not generally possible. It is possible, however, to exhaustively analyze a carefully selected subset of the design space.

Contemporary research in system design has addressed defining the tradespace of a system by 1) focusing on monolithic systems, which can be described primarily by physical parameters (Ross and Hastings 2005; MacCalman 2013; Beery 2016), 2) focusing on SoS, but only considering the physical composition of the SoS (Biltgen et al. 2006; Chattopadhyay 2009), or 3) defining SoS attributes in such an overly simplistic manner that the results do not yield an accurate tradespace (Chattopadhyay 2009). This research aimed to provide a different option by answering the following questions:

- How may the required SoS architectural perspectives of physical, process, and organizational be used to define an SoS design space?
- How may one assess the feasibility of an SoS architecture?
- May the above be used to define an SoS tradespace in an efficient manner so that it can be incorporated into existing MBSE TSE methodologies?

B. RESEARCH AND CONTRIBUTION

1. The SoS Tradespace Definition Methodology

The SoS-TDM is a method to define the tradespace of an SoS according to its physical, process, and organizational parameters. It takes a valid SoS need, relevant performance measures, and potential systems, processes, and organizations as an input and outputs the set of feasible SoS and their performance attributes. It has four steps as seen in Figure 1: “Design Space Definition,” “Design Space Feasibility Analysis and Screening,” “Feasible Design Space Analysis,” and “Design Point Assessment and Tradespace Analysis.”

This methodology is predicated upon the idea that, for any design space, the set of feasible design points is significantly smaller than the entire design space. This is not generally provable, but experience shows it is true in many cases. In particular, as a system increases in complexity, it is generally more difficult to achieve a feasible design, as there are more interactions among the sub-systems, making it more difficult for a system to meet all requirements.

The first step of the SoS-TDM is to define the design space according to physical, process, and organizational parameters. For the physical, this involves defining what systems may be included, what refactorizations[1] they may take, and what communications sub-systems each one has. For the process, this involves defining the potential operational activity flows, defining what functions each system may perform, and defining potential rules of employment. For the organization, this involves defining organizational relationships and the set of organizations that may be formed from these.

The second step of the SoS-TDM is to assess each design point for feasibility. This is done through the SoS-AFAM. In this, each point is assessed as feasible or not according to multiple criteria. The SoS-AFAM is discussed in detail in the next section.

The third step is to assess if the set of feasible design points is "sufficiently small." This is defined as being less than or equal to the number of points that may be assessed in the allowable time. If the set of feasible points is "sufficiently small," then one proceeds to the next step. If the set is not, then one iterates the previous steps at a greater level of detail to further winnow the space.

[1] A refactorization is a slight modification to an existing system. For example, adding a new radio to a vehicle to allow that vehicle to communicate with other systems would be a refactorization.

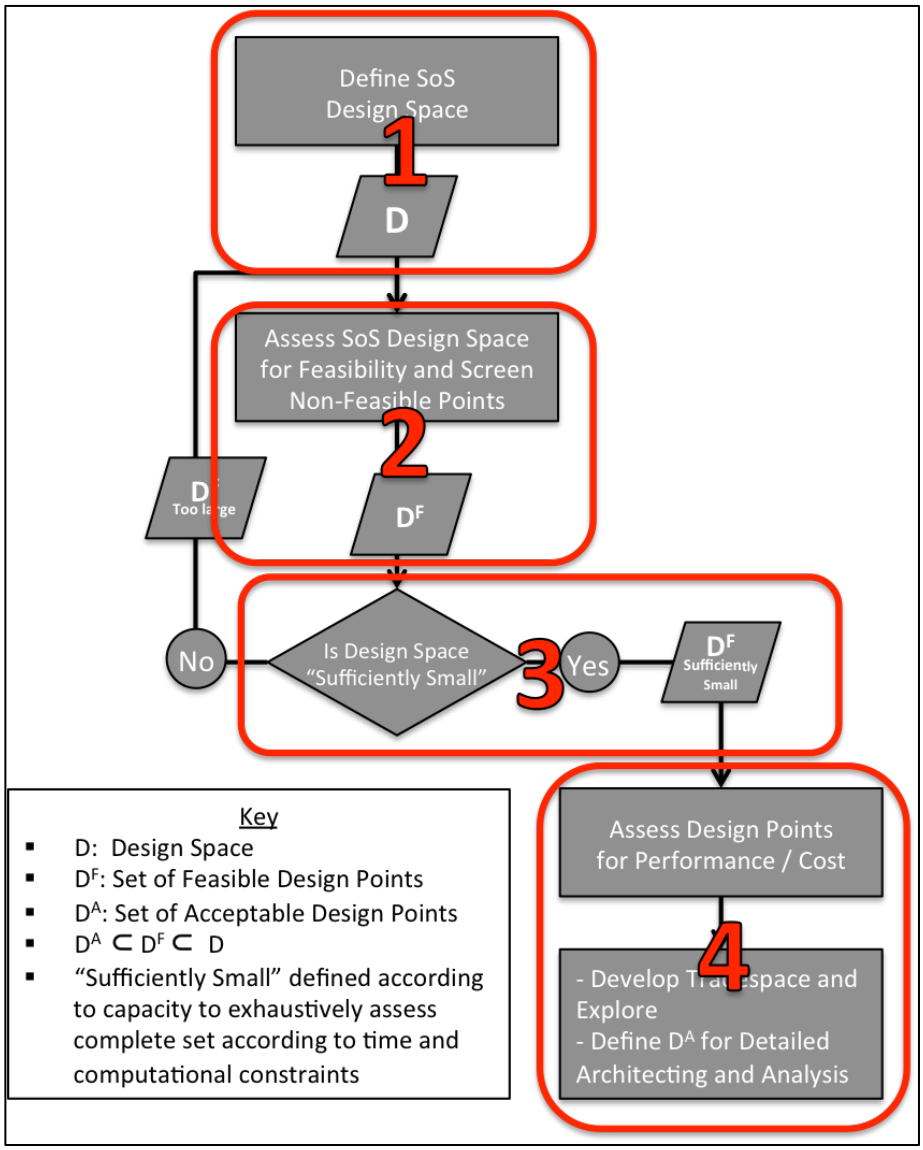


Figure 1. The SoS Tradespace Definition Methodology

The fourth and final step of the SoS-TDM is to assess the design points for their attributes. To do this, one inputs every design point into the relevant model or simulation and records the outcomes using standard techniques. For operational attributes of an SoS , the most common method is through the use of ABM as they best represent the salient aspects of SoS (Rainey and Tolk 2015), although other methods may be used as appropriate. Once one has defined the attributes for each feasible design point, one can build a dynamic visual representation of the tradespace; Figure 2 is an example SoS tradespace visualization.

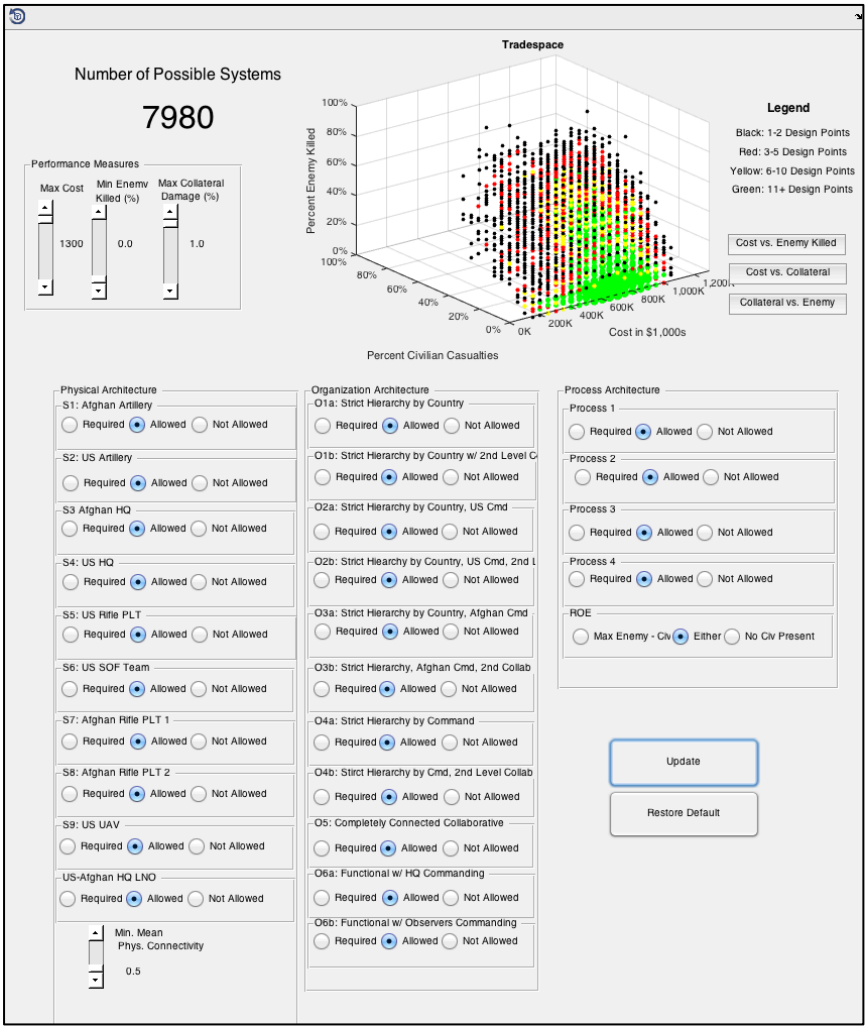


Figure 2. Example SoS Tradespace

A tradespace visualization, as depicted in Figure 2, plots design points according to their attributes, as seen in the top half of the figure (the colors represent the number of design points at each attribute location). One can vary the bounds of the tradespace by imposing requirements for systems, refactorizations, organizations, operational activity flows, or rules of employment to be included or not included in the domain of possible design points. Similarly, one may vary the bounds of acceptable attributes, in this case, cost and performance. In doing this, one varies the set of acceptable design points and "explores" the tradespace. Ultimately, a decision-maker may use this to define a subset of

the feasible design points that are acceptable and then conduct detailed architecting and analysis on these design points and continue the systems engineering process.

2. The SoS Architecture Feasibility Assessment Model

The SoS-AFAM is the second step of the SoS-TDM and depicted in Figure 3. It takes design points as inputs and outputs their feasibility. This is done in four steps where different aspects of the design space are assessed independently. This is advantageous because, by partitioning the design space, one must only assess a small subset of the space, but still be able to comprehensively assess the entire space

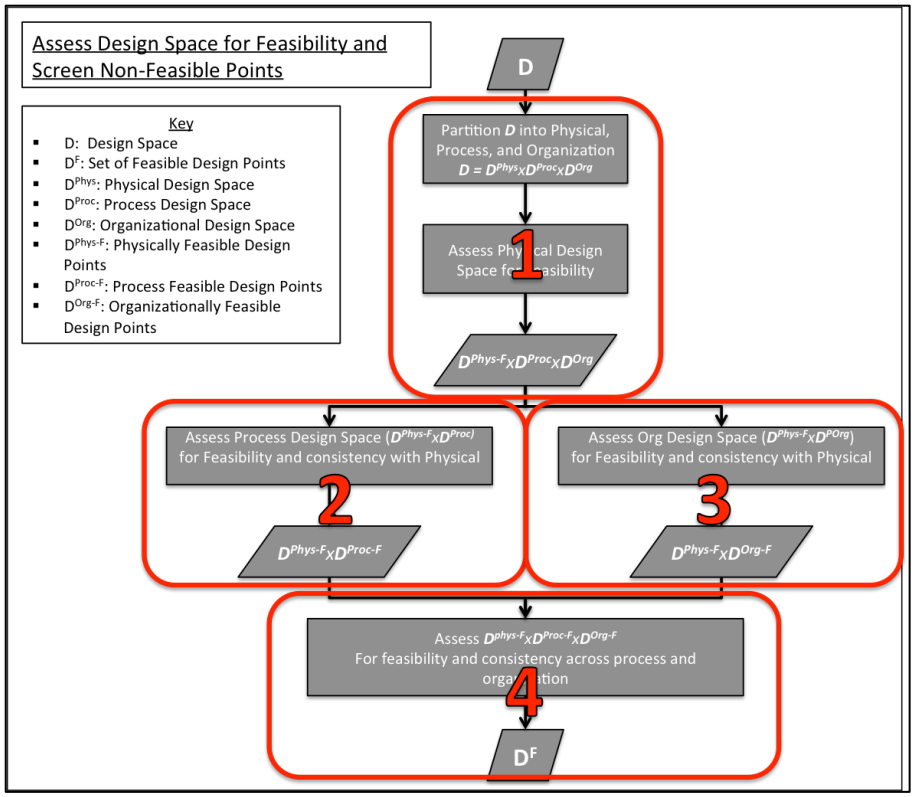


Figure 3. The SoS Architecture Feasibility Assessment Model

The first step of the SoS-AFAM is to assess the physical aspect of all design points. In this step, one assesses each design point's physical parameters against their

ability to form a connected network[2] that is capable of transmitting the required information for that SoS. At a base level, the minimum requirement is that one can form a connected network with the included systems in which a connection between two systems is binary—they are connected if they share a common communications subsystem and not otherwise. At higher levels, one tests for connectivity based upon communications subsystems ranges, availability, minimum bandwidth, maximum latency, and maximum error rate.

The second step of the SoS-AFAM is to assess the process design space. This step assesses every design point composed of a physically feasible set of parameters crossed with all process parameters. The first test assesses if a set of systems has sufficient functionality to complete all functions in the operational activity flow for that point. The second tests assesses if the rules of employment are acceptable to all included systems. The third test assesses if there are any unresolvable conflicts among the constituent systems conducting listed simultaneous activities.

The third step of the SoS-AFAM is to assess the organization design space. This step assesses every design point composed of a physically feasible set of parameters crossed with all organizational parameters. The first feasibility test assesses if the proposed organization is acceptable to all included constituent systems. The second test is if the network formed by the organization (where two systems are connected if they have an organizational relationship) is connected. More detailed tests include acceptance of the number and type of organizational relationships any one system has (e.g., one system may not command more than five other systems), and physical connectivity support for each organizational relationship (e.g., if two systems have a command-subordinate relationship, they must be able to communicate directly).

Finally, one synthesizes the first three analyses to assess which design points are completely feasible. A design point must be feasible from all perspectives—physical, process, and organization. Further, one may assess how well the organization supports the process; in this, one assesses how many organizational steps there are between any

[2] A connected network is one in which every node is connected to every other node either directly or indirectly.

sequential points in an operational activity flow. For example, if one is conducting indirect fire and the activity flow is: observe the target, request fire, and shoot, but the organization between the observer and shooter involves multiple layers of a chain of command, this may not be a feasible solution as the time to traverse the organization may be greater than the allowable time between the two operational activities.

The SoS-AFAM can quickly assess a large design space as it partitions the design space. Specifically, for a given design space, if the number of physical compositions is *C*, the number of processes is *P*, and the number of organizations is *O*, the total number of design points is *CPO*. However, one must only assess a certain percentage of these points; this percentage is

$$\text{Equation 1.} \qquad \Pi = \frac{1}{OP} + \frac{x}{P} + \frac{x}{O} + wx$$

where *x* is the percentage of points that are physically feasible, and *w* is the lesser of the percentage of points that are process or organizationally feasible. Note that as the design space increases in size as a function of organizations and processes, this number decreases. Moreover, the algorithms used to assess each partition of the design space are relatively quick, using common, well-developed network analysis algorithms (e.g. Ahuja et al. 1993).

3. Indirect Fire SoS Example

This dissertation provided an example employment of the SoS-TDM and SoS-AFAM in the development of an indirect fire (IDF) SoS. The IDF SoS is potentially composed of nine systems from four different commands (U.S. Army, U.S. Air Force, U.S. Special Operations, and Afghan Army), one possible refactorization, two possible operational activity flows, two sets of rules of employment, and eleven organizations. This leads to a design space that contains 90,112 design points. Through the use of the SoS-TDM and SoS-AFAM, we identified that we needed a design space with fewer than 10,080 design points to be "sufficiently small." Through the SoS-AFAM, we identified a feasible design space that contained 7,980 points in less than 10 minutes of computation. From there, we developed the SoS tradespace as presented in Figure 2.

C. CONCLUSION

The challenge of designing SoS is a desired but difficult undertaking. SoS are a unique class of systems whose characteristics demand that they be described not only with physical parameters but also with process and organizational parameters that describe how constituent systems interact. One method to facilitate SoS design is TSE; however, contemporary methods of defining tradespaces only consider physical design parameters. SoS designers must address the full complexity of an SoS by including considerations of their relationships—process and organizational parameters. This requirement allows for an extension to the state-of-the-art.

The SoS-TDM and SoS-AFAM extend the state-of-the-art by defining a methodology that winnows a well-defined (physical, process, organization) SoS design space through the use of feasibility tests. This allows one to only assess the feasible design points and use the results to define an SoS tradespace. This tradespace can then be explored and used to define a set of acceptable design points that may then be used for detailed architecting and analysis. The winnowing process, the SoS-AFAM, is a computationally efficient methodology for assessing feasibility for a general SoS. Subsequent research to advance this methodology and model include further development of the models for detailed architecting and analysis; definition and analysis of organizations and processes; the extension of them to collaborative SoS; the extension of the methodology to consider strategic SoS decision making over multiple iterations of the SoS lifecycle; and including environmental considerations to the definition of attributes.

REFERENCES

Ahuja, Ravindra K., Thomas L. Magnanti, and James B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Englewood Cliffs, NJ: Prentice Hall.

Beery, Paul T. 2016. "A Model Based Systems Engineering Methodology for Employing Architecture in System Analysis: Developing Simulation Models Using Systems Modeling Language Products to Link Architecture and Analysis." PhD Dissertation, Naval Postgraduate School.

Biltgen, Patrick T., Tommer Ender, and Dimitri N. Mavris. 2006. "Development of a Collaborative Capability-Based Tradeoff Environment for Complex System Architectures." In *44th AIAA Aerospace Sciences Meeting and Exhibit*, 9–12. doi:10.2514/6.2006-728.

Chattopadhyay, Debarati. 2009. "A Method for Tradespace Exploration of Systems of Systems." Master's Thesis, Massachusetts Institute of Technology.

Department of Defense (DOD). 2008. "Systems Engineering Guide for Systems of Systems." Washington, DC. http://www.acq.osd.mil/se/docs/SE-Guide-for-SoS.pdf.

IDEO. 2016. "About IDEO." Accessed April 5. https://www.ideo.com/about/.

Giachetti, Ronald E. and Clifford Whitcomb. 2016. "Rethinking the Systems Engineering Process in Light of Design Thinking." In *Proceedings of the Thirteenth Annual Acquisition Research Symposium: Volume I:* 48-56. Monterey, CA, May 4-5. Accessed online May 2016 at: https://www.researchsymposium.com/conf/app/researchsymposium/unsecured/file/129/SYM-AM-16-019_Wednesday,%20Vol%201_5-17-2016.pdf.

MacCalman, Alex. 2013. "Flexible Space-Filling Designs for Complex System Simulations." PhD Dissertation, Naval Postgraduate School

Maier, Mark W. 1998. "Architecting Principles for Systems-of-Systems." *Systems Engineering* 1(4), 267–284. doi:10.1002/(SICI)1520-6858

Pernin, Christopher G., Elliot Axelband, Jeffrey A. Drezner, Brian B. Dile, John Gordon IV, Bruce J. Held, K. Scott McMahon, Walter L. Perry, Christopher Rizzi, Akhil R. Shah, Peter A. Wilson, and Jerry M. Sollinger. 2012. *Lessons from the Army's Future Combat Systems Program*. Santa Monica, CA: RAND Arroyo Center.

Rainey, Larry B. and Tolk, Andreas. 2015. *Modeling and Simulation Support for System of Systems Engineering Applications.* Hoboken, NJ: Wiley.

Ross, Adam M., and Daniel E. Hastings. 2005. "The Tradespace Exploration Paradigm." *INCOSE International Symposium*, 15, 1706–1718. doi:10.1002/j.2334-5837.2005.tb00783.x

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

Foremost, I must thank my wife, Theresa, for her love, encouragement, and perspective throughout this process. It has not been easy and I simply could not have done this without her. Also, I would like to thank our son Mason, who was born during our time here in Monterey, for being such a wonderful source of joy—it is a pleasure to watch you grow. I love you both dearly.

I have been fortunate to work with an exceptional committee that has supported, pushed, and developed me and improved this work. In particular, I must thank Dr. Gene Paulo for his vision, guidance, and mentorship. From the very start of this program, he guided and helped me, and set me on the right track. I would like to express my gratitude to Dr. Ron Giachetti for his detailed critiques and insights that drove this work and Dr. Andy Hernandez for helping me to clarify, contextualize, and shape the problem. In addition, I would like to thank Dr. Rudy Darken for his unique and enlightening perspectives. Finally, I would be remiss if I did not acknowledge Mr. Paul Beery for his ever-patient explanations of complex subjects and willingness to listen to and vet my ideas. I thank you all for making this an enriching and rewarding experience—you have taught me much and I deeply appreciate it. You are all academics and gentlemen of the first order, and I hope to work with you again in the future.

I am grateful for the education, training, and professional development I have received at the Naval Postgraduate School, and in particular, at the Department of Systems Engineering. This is a testament to the faculty, staff, and students—thank you. Finally, I owe a significant debt of gratitude to the U.S. Military Academy Department of Systems Engineering and Colonel Robert Kewley who took a chance and offered me the opportunity for this education. I hope I can repay the investment in my service to the department and the cadets it educates.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

This dissertation contributes to the state-of-the-art in two sub-fields of systems engineering, System of Systems Engineering (SoSE) and Model-Based Systems Engineering (MBSE). Current methods of designing SoS are either 1) heuristic or 2) analytic and focus on the physical considerations of an SoS while neglecting the process and organizational ones; however, these considerations are necessary as they represent how an SoS provides its capabilities. Furthermore, MBSE design methodologies are challenged to address the problem of accounting for process and organizational considerations, as they have been developed explicitly for monolithic systems. This dissertation contributes the SoS Tradespace definition Methodology (SoS-TDM) and SoS Architecture Feasibility Assessment Model (SoS-AFAM) as a means to add considerations of process and organization to the design of an SoS.

The SoS-TDM is predicated upon several basic concepts. First, tradespace exploration significantly facilitates system design and augments heuristic and normative decision-making methods. Second, for accurate analysis, an SoS design space must be defined by physical, process, and organizational parameters; this inherently expands the size of the design space. Third, all SoS design points may be assessed quickly, in a general manner, for feasibility through the use of the SoS-AFAM. Finally, the subset of the SoS design space that is feasible is significantly smaller than the entire design space. With sufficient feasibility analysis, an engineer may winnow the design space to a sufficiently small set of feasible solutions that may be analyzed exhaustively. The result of these concepts is that an engineer can define an SoS design space that includes parameters necessary to define an SoS architecture, winnow this design space by only considering the feasible design points, and then assess these points for performance attributes and build a tradespace for subsequent exploration and analysis.

A. MOTIVATION AND BACKGROUND

A SoS is composed of multiple operationally and managerially independent systems that interact to produce a desired capability not provided by any individual

system. Moreover, the design and operation of an SoS is not wholly controlled by any one entity. Organizations—governmental, private, and combinations thereof—increasingly rely upon SoS to meet their needs. This is due, in part, to the networked nature of modern society and to a recognition that SoS are capable of meeting needs that monolithic systems either cannot meet or are inefficient in meeting.

In particular, the Department of Defense (DOD) is interested in SoS design and development as it owns and operates multiple SoS and foresees developing future SoS. Examples of former and current SoS the DOD owns or is a part of include the Army's Future Combat System, the Navy's Naval Integrated Fire Control – Counter Air, the Air Force's Air Operations Center, and the Joint Ballistic Missile Defense System (Department of Defense [DOD] 2008, 2–3). The Navy's concept of "Distributed Lethality" that proposes forces composed of multiple distinct interoperating systems to provide a new, greater capability (Rowden et al. 2015) is a future Navy SoS. The Army's Operating Concept, "Win in a Complex World" establishes a need to provide "effective integration of military, interorganizational, and multinational efforts" (U.S. Army 2014, iv). In short, the Army's concept is to be able to quickly develop SoS including U.S. Army, joint, and other forces to contend with emergent situations. To address how the DOD designs, acquires, and manages SoS, it has published the "Systems Engineering Guide for Systems of Systems" (DOD 2008). The DOD clearly has an interest in designing SoS that meet its stakeholders' needs.

Designing SoS that successfully meet stakeholders' needs has proven to be a difficult undertaking. The Army's Future Combat System (FCS) is an example of an SoS design failure. It suffered for lack of clear SoS level architecting and analysis (Pernin et al. 2012, xx-xxiii) and a "narrow level of focus at the program level rather than at the level of the enterprise" (Archer 2014, 23). Furthermore, there were, "conflicts of interest among the different stakeholders of the project and an inability to observe these conflicts easily" (Srivastava, Piper, Arias 2012, 1964). More specifically, Pernin et al., (2012) in a RAND Corporation analysis of the FCS program identified the following best SoSE practices the FCS program failed to employ:

- Analytic capabilities are important to the success of large, complex acquisition programs. The development of concepts and the analysis of cost, technical feasibility, risk, and uncertainty all require detailed and sophisticated study.
- An organization and operation (O&O) plan that takes an integrated unit perspective can aid requirements formulation.
- A successful program requires a sound technical feasibility analysis.
- The development of operational requirements requires an integrated, unit-level (not system-level) approach
- Up-front system engineering and architecting are critical
- A shared modeling and simulation repository can improve the fidelity of mission-level analysis. (Pernin et al. 2012, xviii–xxix)

Pernin et al. (2012) specifically note that the analysis of SoS technical feasibility, organization, and operations are key to SoSE. The FCS program either did not do these or did them poorly, and, consequently, the FCS failed to materialize. This failure was costly at $14 billion (2012 U.S. dollars) and 10 years of effort (Pernin et al. 2012, 50). A SoS design methodology that addresses these issues—the need to assess for feasibility, include organization and operations in the architecture, and conduct up-front SoSE—would improve the ability of organizations to design and realize successful SoS.

Coincident with the challenge of and necessity for SoS design have been advancements in the field of MBSE, particularly as it relates to design decision-making, namely tradespace exploration (TSE). Many researchers have examined tradespace exploration in the context of MBSE, e.g., (Brantley et al. 2002; Stump et al. 2005; Ross 2006; Carlsen 2008; Chattopadhyay 2009; Sitterle et al. 2015; Beery 2016; Paulo, Beery, and MacCalman forthcoming). In particular, Beery (2016) developed the MBSE Methodology for Employing Architecture in System Analysis (MEASA) that formalizes a linkage with MBSE architecture description models and analysis models (Beery 2016). This methodology has proven useful for facilitating design decision-making for singular systems. An expansion to the MEASA or other similar system design methodologies for SoS will facilitate improved SoS design decision-making.

Many organizations desire to engineer SoS to meet their needs, particularly the DOD. Designing and realizing an SoS has proven difficult and resulted in costly failures. This is, at least in part, because the design of an SoS must account for unique SoS considerations. Consequently, there is significant utility in developing methodologies and tools that facilitate and improve SoS design.

B. SYSTEM DESIGN AND DECISION-MAKING

This dissertation considers design as the process of determining a system architecture. It is necessarily an iterative process between the creative act of imagining possibilities and the analytic act of assessing those possibilities for feasibility and other performance measures (Cross 2011, 16–29; Buede 2000, 37–41). This inherently involves decision-making—what the system must do, how it may do it, and how well it must do it. There are at least three general methods of decision-making: heuristic, normative, and exploratory.

Heuristic decision-making is founded in principles based upon experience and best practices. Maier and Rechtin (2009) outline an extensive number of systems architecting heuristics. Within the field of SoS, Maier (1998), Cole (2008), and Dagli and Kilicay-Ergin (2009) outline various heuristics. Heuristics are useful, but are limited as they are often conflicting (as they apply in varying contexts) and only provide general guidance. Moreover, heuristics must be applied by a knowledgeable designer.

Normative decision-making is founded upon making decisions for a well-defined, well-understood problem. Clear performance measures and their associated values are defined and engineers make decisions based upon optimizing these measures. This is commonly practiced in traditional systems engineering (Keeney 1992; Buede 2000; Blanchard and Fabrycky 2011; Parnell, Driscoll, and Henderson 2011). Normative design has also been called "technical rational design;" Giachetti and Whitcomb (2016) clearly articulate its baseline assumptions and its benefits and limitations. This is useful in many cases, but less so when the understanding of the problem and potential solutions is poorly understood. In fact, the premise of normative decision-making is that stakeholders' values exist independently from the problem and must only be "elicited," whereas

psychologists have identified that preferences are often "constructed" (Lichtenstein and Slovic 2006). Accordingly, it is often useful to use exploratory analysis to better understand and define decision-maker requirements and values.

The final decision-making methodology is exploratory. This is, in essence, trial and error—closely coupled iteration between solution definition and analysis. This takes many forms, it is sometimes called, generically, "design thinking" (Cross 2011; Giachetti and Whitcomb 2016), but more rigorous implementations of it come in the form of tradespace exploration (TSE).

While there is no definitive definition of a tradespace, the term is used extensively in the literature (Brantley et al. 2002; Stump et al. 2005; Ross 2006; Carlsen 2008; Chattopadhyay 2009; Sitterle et al. 2015; Beery 2016; Paulo, Beery, and MacCalman forthcoming). The general concept of a tradespace is based upon the idea that, for any system design problem, there is a design space. The design space is the set of all possible system design points, described by system parameters. Each design point has system attributes that describe how the system performs (e.g., operational performance, cost). The tradespace is the combination of the design space and the space defined by the system attributes; these spaces vary in size and composition depending upon constraints decision-makers place upon what system parameters and system attributes are acceptable and desirable. As these spaces vary depending upon decision-maker requirements, one may identify and understand the trade-offs involved in any threshold requirement, attribute value, or weighting of attributes, hence the name tradespace.

Until recently, the concept of a tradespace was, by and large, theoretical; it was difficult to define and explore the tradespace in any meaningful manner. However, advances in computational power, statistical methods, and MBSE tools and methodologies have made tradespace definition and exploration a third possibility for system design.

C. MODEL-BASED SYSTEMS ENGINEERING AND TRADESPACE EXPLORATION

The International Council on Systems Engineering (INCOSE) defines MBSE as "the formalized application of modeling to support system requirements, design, analysis, verification and validation, beginning in the conceptual design phase and continuing throughout development and later life cycle phases" (Friedenthal et al. 2007, 5). Although it has broad applications, MBSE has particular impact upon the design of systems. This is because skillful employment of MBSE broadens the ability of an engineer to develop, understand, and assess significantly more alternative options in a system design problem, this "illuminates the tradespace" (Paulo, Beery, MacCalman forthcoming).

There has been much research regarding tradespace exploration in a MBSE environment (Stump et al. 2004; Ross and Hastings 2005; Sitterle et al. 2015; MacCalman et al. 2015; Beery 2016; Paulo, Beery, and MacCalman forthcoming). In particular, recent research (Beery 2016) has defined a useful methodology that uses MBSE tools to define the tradespace for a system using systems architecture models. Beery's (2016) MBSE MEASA advanced the state-of-the-art in MBSE by explicitly integrating systems architecture models with analysis models to allow for subsequent exploratory design. This was intended for systems with the assumption of top-down, monolithic design. This assumption is invalid for SoS as they are developed "bottom-up," meaning that the constituent systems execute a level of independence. Moreover, the MBSE MEASA does not consider non-physical factors such as process or organization (Beery 2016). These perspectives are, however, important in the design of an SoS.

D. SYSTEMS OF SYSTEMS ENGINEERING AND DESIGN

SoS are a unique subset of systems that require special consideration and architectures that direct and describe how the constituent systems interact in order to provide useful capabilities (Maier 1998; Dagli and Kilicay-Ergin 2009). In particular, one may vary the process and organizational architecture aspects of an SoS while holding the physical architecture constant and produce different capabilities, both in degree and kind.

The architecture of an SoS , therefore, must include these perspectives. To fully explore the wide variety of potential SoS, one must define a design space that incorporates the parameters that describe these architectural requirements.

1. System of Systems

A SoS is commonly defined as a system composed of multiple systems that are operationally and managerially independent, geographically dispersed, present emergent behavior, and develop in an evolutionary manner (Maier 1998). Other authors have varied the criteria, e.g., autonomy, belonging, connectivity, diversity, and emergence (Boardman and Sauser 2006) or Maier's five characteristics plus self-organization and adaptation (Sage and Biemer 2007). Regardless of the precise definition, a general consensus is that an SoS is a system, composed of multiple independent systems, that provide some capability and the total design of the system is not wholly controlled by any one entity.

The DoD has adopted a classification of virtual, collaborative, acknowledged, or directed SoS (DOD 2008). The classification distinguishes SoS based on the amount of managerial control the SoS level has, with virtual and collaborative SoS having none to minimal, acknowledged having limited, and directed having significant control. They further distinguish SoS based upon the agreement of the SoS's purpose, with virtual SoS having no agreement and the others having an agreed upon purpose for the SoS.

SoS provide a capability that is not wholly encapsulated by any one system. This capability is a product of the interactions that occur among the various constituent systems, typically called an emergent behavior. Emergence may be very simple and predictable, such as gears rotating in a watch to keep time or highly complex, such as neurons in a brain yielding consciousness (Maier 2015). All systems, SoS or otherwise, exhibit emergent properties; however, SoS are distinct in that the designer of an SoS does not completely control how the constituent systems (i.e., its sub-systems) are designed, how those systems function, or how those systems operate. The challenge for an SoS engineer is to design an SoS that will cause the constituent systems to interact in a productive manner. These interactions are founded upon the physical systems included in the SoS and the rules that guide their behavior, the process and organizational architectures.

2. System of Systems Architecture

An architecture prescribes a system's structure in terms of elements and relationships from multiple perspectives. A standard trichotomy of systems architecture is functional, physical, and allocated architectures (Buede 2000). A functional architecture describes *what* a system must do. The physical architecture represents how the system is physically partitioned, colloquially, the *who* of the system. The allocated architecture maps the *who* to the *what.* Finally, architectures may be standardized using architecture frameworks such as the Zachman Framework or DOD Architecture Framework (DODAF). For this dissertation, DODAF is used as a frame of reference, although the approach is generally applicable.

Within an SoS , the physical-functional-allocated trichotomy is valid, but there are some key distinctions. At its highest level, the functional architecture of an SoS represents, in part, the emergent properties of the SoS, *what* the system must do to provide its useful capabilities. The physical architecture of an SoS describes included constituent systems that are, generally, pre-existing to the SoS. The allocated architecture of an SoS is very distinct from general monolithic systems. Standard engineering practice is to allocate functions to physical sub-systems in a one-to-one manner (Buede 2000). For monolithic systems, this works as engineers have control over the development of their sub-systems and development is "top-down." In an SoS , this is generally not true. The SoS designer does not have control over the development of the constituent systems and the development process is "bottom-up." Moreover, different constituent systems may have the capacity to provide the same functions. SoS must describe how constituent systems interact and are "assigned" to functions. This is commonly expressed as process and organizational architectures.

For this dissertation, an SoS physical architecture describes the composition of the included constituent systems and the communications network formed by these systems. At its highest level, it is a graph (network) where the nodes represent the constituent systems and the arcs represent communications links. At lower architectural levels, the details of the constituent system capabilities, communications standards, communications systems performance, and other similar detail are included in this architecture. Though this architecture may be expressed in multiple ways, the DODAF describes this primarily

in through the Operational Viewpoint 1 (OV-1), high-level operational concept, Systems View-1 (SV-1), system interface matrix, and the Data and Information Viewpoints (DIV) (Department of Defense Chief Information Officer [DOD CIO] 2010).

The process architecture describes both the operational activity flow (expressed as a kill chain, functional flow block diagram, IDEF0 diagram, or similar flow model) and the rules of employment that govern this flow. Though these may be expressed in different ways, the DODAF describes this in its various Operational Viewpoints (OV) and certain System Viewpoints (DOD CIO 2010).

Finally, the organization architecture describes the relationships between the constituent systems. This includes both a definition of the relationships with regard to how they affect system decision-making (e.g., one system prioritizes a response to another system due to a hierarchical relationship between the two) and what information is required, permitted, or prohibited between two relationships. This is seen in DODAF in the OV-4: Organizational Relationships Chart and may be seen in variations of the aspects of the Services or Systems Viewpoints (DOD CIO 2010).

SoS architecture descriptions may be done using many of the same tools and methods for describing monolithic systems. Pan, Yin and Hu (2011) demonstrate the utility of modeling and simulation of SoS using DODAF. DODAF 2.02 makes provisions for SoS. Similarly, MBSE tools such as SysML are useful to represent SoS (Lane and Bohn 2013; Wang 2007; Rao et al. 2008; Kenley et al. 2014). It is important to note, however, that within these frameworks, methodologies, and tools, engineers must take care to specifically identify the important SoS aspects of the physical, process, and organizational views as, together, these views describe and prescribe how the constituent systems interact to provide desired SoS capabilities.

3. System of Systems Analysis

Once an SoS architecture has been described, it must be analyzed for its performance attributes (e.g., feasibility, cost, operational performance). SoS analysis differs from typical systems analysis (Buede 2000; Blanchard and Fabrycky 2009; Gibson et al. 2007) primarily in the details. Notably, it differs in what is being analyzed and the tools used to assess SoS. The purpose of SoS analysis is to assess how an SoS performs

according to any number of measures of effectiveness (MOE) or measures of performance (MOP). These measures should focus on the desired emergent capabilities provided by the SoS (Thompson et al. 2015). To assess these emergent properties, engineers are best served using models that demonstrate them. This is most commonly expressed through the use of Agent Based Models (ABM) (Rainey and Tolk 2015), through Petri Nets (Wang 2007; Rao et al. 2008; Kenley et al. 2014), Markov Chains (Giachetti 2015), and other simpler aggregation models (Chattopadhyay 2009).

Within any systems analysis, particularly in the context of tradespace development, one must assess large numbers of design points. Due to the nature of SoS, to accurately assess them for performance, one must represent their complex interactions, at least across the physical, process, and organization perspectives. Given that time and computational power are finite resources, it makes sense to only assess carefully selected design points. Logically, it only makes sense to assess the design points that have the potential to be realized, the set of feasible points. An efficient feasibility test that assesses an SoS design point against feasibility requirements from multiple perspectives allows one to winnow the design space and exhaustively examine the significantly reduced sub-set of feasible design points.

Finally, note that in modeling a system (of any sort), one must identify the relevant interactions. The identified interactions must be known *a priori* to do this. It is possible that there are interactions that are not foreseeable, no matter how carefully one considers and understands the problem; this is an inherent limitation of modeling and simulation. On the other hand, many, if not most, interactions are foreseeable, even if they were not actually foreseen. The art of modeling and simulation involves scoping a modeling problem so that one sufficiently identifies the most relevant interactions to correctly approximate the behavior of the system. For SoS, in addition to baseline physical concerns, considerations of organization and process are relevant and significantly contribute to the interactions that lead to emergent behavior. It is impossible to say that all interactions will occur from only a physical, process, or organization perspective; however, many, if not the majority of SoS interactions may fall into these categories.

4. System of Systems Design

SoS design methodologies, tools, and guidance come in the form of heuristics, normative decision-making, and exploratory decision-making methodologies. The most significant reference is Maier's (1998) SoS architecting principles (heuristics). Other methods have been proposed and Figure 1 outlines them. These various methodologies are generally limited, however, in that considerations of SoS-specific architecture requirements of organization and process are either not, or poorly accounted for. In particular, the two SoS TSE specific methodologies, Chattopadhyay (2009) and Biltgen et al. (2006) are similarly insufficient. Chattopadhyay (2009) only considers SoS composition. Biltgen et al. (2006) is focused on physical interactions of sub-systems within a system or directed SoS.

The other SoS research does not account for tradespace exploration. In particular, Rao et al. (2008) focused on integrating SysML with Petri Nets; Mokhtarpour and Stracener (2014) is limited and does not consider the requirements for organizational and process architecture; Davendralingam and DeLaurentis (2015) provides methods for considering the different combinations of systems, but they are focused on optimizing pre-established metrics and not tradespace exploration. Kenley et al. (2014) is the most closely related research; it includes allocation of systems to functions, but in a very limited manner and it does not assess for SoS feasibility (Kenley et al. 2014).

5. System of Systems Conclusion

SoS are a distinct subset of systems with unique architecture, analysis, and design requirements. In particular, for accuracy and completeness, SoS architectures require a description of their physical, process, and organizational perspectives. This significantly impacts subsequent SoS analysis and operational performance. Accordingly, to explore an SoS tradespace, the design space must include these parameters. This has not been done in the field of tradespace exploration and poses a potential extension to the state-of-the-art. The ability to define and analyze an SoS design space efficiently allows the development of an SoS tradespace, which provides engineers and analysts a third tool for SoS design decision-making.

E. CONCLUSION

There is a significant need to design SoS; however, this is a difficult challenge. SoS must be designed in a manner that includes their physical, process, and organizational considerations. These have been expressed in SoS architectures and included in SoS heuristic design decision-making. They have not been included in more analytic SoS design techniques, particularly TSE, as seen in Figure 1. Furthermore, by including expanded SoS design parameters, we challenge existing methods to account for the complex interactions among these various parameters. We must, therefore, introduce a different methodology for defining and exploring the tradespace.

		Design Decision Making Methodology		
		Heuristics Decision Making	**Normative Decision Making** **Traditional SE**	**Exploratory Decision Making** **Design Theory / Tradespace Exploration**
Classification of Systems	**Monolithic Systems**	• *The Art of Systems Architecting,* Maier and Rechtin (2009)	• *The Engineering Design of Systems,* Buede (2009) • *Decision Making in Systems Engineering and Management,* Parnell and Driscoll (2011) • *Systems Engineering Analysis,* Blanchard and Fabrycky (2011) • *Defense Acquisition Guidebook Chapter 4, Systems Engineering*, DOD (2013) • *Systems Engineering Handbook,* NASA (2014) • *Systems Engineering Handbook*, INCOSE (2015)	• "Design Space Visualization and Its Application to a Design by Shopping Paradigm," Stump et al. (2004) • "The Tradespace Exploration Paradigm," Ross and Hastings (2005) • "Systems Engineering Resiliency: Guiding Tradespace Exploration within an Engineered Resilient Systems Context," Sitterle et al. (2015) • "Illuminating Tradespace Decisions Using Efficient Experimental Space-Filling Designs for the Engineered Resilient System Architecture," MacCalman et al. (2015) • "A Model-Based Systems Engineering Methodology for Employing Architecture in System Analysis," Beery (2016)
	Systems of Systems	• "Architecting Principles for Systems-of-Systems," Maier (1998) • "SoS Architecture," Cole (2008) • "System of Systems Architecting," Dagli and Kilicay-Ergin (2009)	• "A Robust Portfolio Optimization Approach to SoS Architectures," Davendraglingam and DeLaurentis (2015) • "A Conceptual Methodology for Selecting the Preferred SoS," Mokhtarpour and Stracener (2014) • "Synthesizing and Specifying Architectures for SoS," Kenley et al. (2014) • "Modeling and Simulation of Net Centric System of Systems Using Systems Modeling Language and Colored Petri-Nets," Rao et al. (2008) • "Systems Engineering Guide for Systems of Systems," DOD (2008)	• **"The System of Systems Tradespace Definition Methodology Through the System of Systems Architecture Feasibility Assessment Model," Gillespie (2016)** • "A Method for Tradespace Exploration of SoS," Chattopadhyay (2009) • "Development of a Collaborative Capability Based Tradeoff Environment for Complex System Architectures," Biltgen et al. (2006)

Figure 1. Design Decision-Making References by Methodology and System Type

This leads to a potential extension to the state-of-the-art in both MBSE and SoSE. The extension is in adding the perspectives of process and organization to existing TSE methodologies. By adding these new considerations, we must, however, be able to define SoS feasibility from these multiple perspectives, as any chosen design point must be feasible. Assessing for feasibility allows us to define a small sub-set of the entire design space for exhaustive analysis.

This research addresses these potential extensions by answering the following problems:

- How may the required SoS architectural perspectives of physical, process, and organizational be used to define an SoS design space?
- How may one assess the feasibility of an SoS architecture?
- May the above be used to define an SoS tradespace in an efficient manner so that it can be incorporated into existing MBSE TSE methodologies?

The scope of this research is limited to studying acknowledged and directed SoS. It is focused on SoS design, in particular, high-level, early life-cycle design and architecture. Furthermore, it is limited to the bottom-up design of SoS composed of existing systems.

The end state of this research is two-fold. First, it is a general methodology, the SoS-TDM, to describe a means of defining and examining the tradespace of SoS in a manner that includes parameters that describe the SoS physical, process, and organizational architectures. Second, it is a specific modeling technique, the SoS-AFAM, to assess SoS feasibility using the same parameters. The results may be used in conjunction with a greater MBSE TSE approach and/or SoS engineering methodology.

THIS PAGE INTENTIONALLY LEFT BLANK

II. LITERATURE REVIEW

This chapter introduces the relevant background material, defines key terms and concepts, and discusses recent, related research. This provides readers with a common language and the context in which the research provides an original contribution. For clarity and brevity, the author assumes the reader has familiarity with the foundations of systems engineering.

In particular, this chapter outlines system design and decision-making, tradespace exploration, MBSE, and SoS engineering and design. Together, these areas show a potential extension to the state-of-the-art in both MBSE and SoSE as applied to designing SoS. This is because current SoS exploratory design methodologies do not allow for the architecture requirements of process and organizational views.

A. SYSTEM DESIGN AND DECISION-MAKING

> Design is the essence of engineering. Broadly defined, design is the creative process by which our understanding of logic and science is joined with our understanding of human needs and wants to conceive and refine artifacts that serve specific human purposes. (White 1998, 285)

The focus of this dissertation is SoS design. White (1998) provides a useful definition of design that exemplifies what others (e.g., Buede 2000; Blanchard and Fabrycky 2009; Maier and Rechtin 2009) have stated about design (or architecting): it is an iterative process that necessarily combines creativity, analysis, and judgment and a balancing act of satisfying multiple, possibly competing, requirements and constraints. As one chooses among the range of possible problem definitions and system solutions, an engineer must have a "rational, explicit process" (Buede 2000, 13) that facilitates this decision-making.

Broadly speaking, there are three major methods of design decision-making: heuristic, normative, and exploratory. The first two are broadly explored in the literature; the latter is less well documented, and may be termed "design thinking" or "tradespace exploration" depending upon the context. There is no specific ordering to these

methodologies, each has its strengths and weaknesses; the three methodologies are generally complementary.

1. Heuristic Decision-Making

A heuristic is "A guideline for architecting, engineering, or design. Lessons learned expressed as a guideline. A natural language abstraction of experiences that passes the tests of Chapter 2"[3] (Maier and Rechtin 2009, 424). Simply put, a heuristic is an expression of common sense experience. These are highly useful in systems design. They provide guidelines to reduce ambiguity, contend with complexity, and facilitate decision-making when means that are more analytic are not feasible. Moreover, they are very quickly employed and can be used to find a "good" solution in reasonable time (Giachetti 2010).

Maier and Rechtin (2009) compiled a significant number of systems architecting heuristics. Within SoSE, Maier (1998) and Cole (2008) have proposed several heuristics. These are

- Maier (1998): "Stable Intermediate Forms," "Policy Triage," "Leverage at the Interfaces," and "Ensuring Cooperation,"
- Cole (2008): "Needs often compete," "Needs change over time," "Resource availability constrains the solution space," and "Design compromise is necessary."

Heuristics do have their limitations: they sometimes conflict; they may be victims of experience; and they have difficulty in providing guidance for choices that vary in degree. First, heuristics may provide contradictory guidance. Maier and Rechtin (2009) provide an example that "Look Before You Leap" and "He Who Hesitates is Lost" are, 1) obviously contradictory and 2) situation dependent. They get around this by defining heuristics as narrowly focused on a single field, although this may not prevent all contradictions. Second, heuristics, to be effective, must ring true with the decision-maker;

[3] The "tests of Chapter 2" are "There is an interesting human test for a good heuristic. An experienced listener, on first hearing one, will know within seconds that it fits that individual's model of the world. Without having said a word to the speaker, the listener almost invariably affirms its validity by an unconscious nod of the head, and then proceeds to recount a personal experience that strengthens it. Such is the power of the human mind" (Maier and Rechtin 2009, 31).

accordingly, this is subject to that decision-maker's personal bias and experiences. Adams (2001, 70), writing on creativity, states, "The problem arises when individuals become so universally in favor of tradition that they cannot see the need for and desirability of change in specific areas." The choice and employment of a heuristic is subject to this challenge. Finally, when making decisions among options that vary in degree (as opposed to kind), heuristics are limited, as distinguishing between the degrees of options requires analysis. Together, these limitations lead to the fact that an experienced and skillful designer must employ heuristics. In cases where these limitations are apparent, other decision-making methodologies are useful.

2. Normative Decision-Making

Normative decision-making is the typical analysis expressed in most systems engineering and analysis texts. It is also sometimes termed "technical-rational design" (Giachetti and Whitcomb 2016). It involves defining a problem through significant interaction with stakeholders, establishing metrics by which to assess solutions, defining value curves that normalize the metrics and clarify the importance stakeholders place upon various solutions, and defining relative weights among the metrics (Buede 2000; Parnell, Driscoll, and Henderson 2011; Blanchard and Fabrycky 2011). With this framework in place, a problem is well defined and potential solutions may be analyzed against these metrics. Once a set of potential solutions are defined and analyzed, one may establish a set of Pareto optimal solutions, among which the decision-makers must choose.

This type of decision-making is very powerful. It allows engineers and analysts to quantify various options and weigh them against each other. It provides a means of limiting subjectivity in decision-making and helps inform decision-makers of how various options perform. Due to the success of normative decision-making, particularly for problems that are well defined and easily quantified, this sort of decision-making is pervasive in many industries.

This type of decision-making is also limited. It is subject to the bias of initial problem definition—requirements (e.g., thresholds and goals on various measures) and

values may be defined incorrectly. It presupposes that decision-makers have intrinsic values that may be "elicited;" we must only interrogate the stakeholders sufficiently to understand these preferences. However, psychologists have recognized that preferences are often "constructed," i.e., preferences are often developed in the context of a situation (Lichtenstein and Slovic 2006). In the field of systems engineering, this manifests itself when a system is designed such that it meets all of its stated requirements, yet stakeholders are, ultimately, unsatisfied. Norman and Kuras (2006, 207) articulate this clearly:

> We continue to view Systems Engineering as fundamentally about allocating desired, known functionality among specific elements of a design; all known a priori and stable over time. The users of the functionality built often accuse us, the developers and acquirers, of being "late to need," "unresponsive," and "too expensive."
>
> We respond with a lexicon carefully crafted to put the onus back on the users. We say that the users' requirements are unknown or poorly stated; that, if the requirements are known, there is a requirements drift (i.e., modifying the requirements), or requirements creep (i.e., adding additional requirements). We suggest that the user can't (or won't) say what they really want, or how they will use that which is to be built and delivered. (Norman and Kuras 2006, 207)

This problem leads to one of a few possible solutions. Decision-makers increase the number of requirements in an attempt to better define what they desire (leading to a reduced possible design space) or decision-makers return to heuristics or, worse, personal bias. An alternative to these options is exploratory decision-making.

3. Exploratory Decision-Making

The final general methodology for decision-making is exploratory. This is a non-standard term, but encompasses related ideas seen throughout the literature. For this dissertation, exploratory decision-making is defined as closely coupled iteration of synthesis and analysis. This broadly encompasses seemingly distinct methodologies as "design thinking" and "tradespace exploration."

Companies such as IDEO popularized "Design Thinking." Tim Brown, the president of IDEO, defined it: "Design thinking is a human-centered approach to

innovation that draws from the designer's toolkit to integrate the needs of people, the possibilities of technology, and the requirements for business success" (IDEO 2016). Important to this is the idea that there are overlapping requirements for system design that consider desirability and feasibility and that solution formulation is not an orderly process, rather a sequence of "inspiration, ideation, and implementation" (IDEO 2016). One does this through the development and trial of prototypes or similar models of the solution. Other authors have expanded upon the concepts of design (Cross 2011; Nelson and Stolterman 2003; Whitcomb and Giachetti 2016). These are by and large theoretical (and, in some cases philosophical) constructs of design thought. Design is useful as it explores both problem definition and solution simultaneously.

A related concept to exploratory decision-making is "set-based design" or "set-based concurrent engineering." This methodology was most prominently employed by Toyota and discussed by Sobek, Ward, and Liker (1999). The general concept is that throughout the design process various domain engineers (e.g., mechanical, manufacturing) and other perspectives (e.g., marketing) consider the set of all possibilities, gradually eliminating infeasible solutions (Sobek, Ward, and Liker 1999). This is in contrast with traditional engineering, in which engineers attempt to converge on a (optimal) point. In the case of Toyota, this method is particularly useful as it is tied to their product development process (Sobek, Ward, and Liker 1999). From a defense perspective, there has been some application to naval engineering (Singer, Doerry, and Buckley 2009; Doerry et al. 2014). In particular, this has been applied to early stage capability development conceptualization for an amphibious combat vehicle (Doerry et al. 2014). To date, these applications have been for monolithic systems.

More analytically, various researchers developed the concept of tradespace exploration (TSE) to address similar problems seen in normative decision-making. The essence of TSE is that a tradespace is a design space composed of potential design points and their associated performance measures (this is more rigorously defined in the next section). Through this, designers and decision-makers can explore their options both in terms of system design and system performance. This concept has been explored and developed by a wide variety of researchers (Stump et al. 2004; Ross and Hastings 2005;

Pennsylvania State University Applied Research Laboratory [PSU-ARL] 2015; Sitterle et al. 2015; MacCalman et al. 2015; Beery 2016; Paulo, Beery, and MacCalman forthcoming). In particular, the Pennsylvania State University Applied Research Laboratory Trade Space Exploration Group (2015) defines it as a "shopping process," "negotiated process," and "iterative process." TSE allows engineers to use analytic tools to develop virtual design spaces that may be used in a "design thinking" manner as outlined by IDEO (2016). By virtue of being composed of computer models, researchers may consider increasingly complex or cost-prohibitive (for proto-type development) solutions that would normally be done in a non-analytic design-thinking environment.

Exploratory decision-making is a third option to augment heuristic and normative methodologies. It provides flexibility in problem definition (a problem in normative methods) while allowing for analytic comparisons (a problem in heuristic methods). This augments the other methods and facilitates high-level design decision-making and allows users to better formulate problems (using their experience and heuristics) and requirements for subsequent optimization.

4. Conclusion

The design of a system involves decision-making. In general, there are three general decision-making methodologies: heuristic, normative, and exploratory. Each has its own benefits and limitations; the three augment each other and should be used in combination for any full system design problem. The third method, exploratory, is the most recent as advances in computer modeling and simulation have made large-scale tradespace exploration feasible.

B. TRADESPACE, TRADESPACE EXPLORATION, AND DESIGN DECISION-MAKING

Exploratory decision-making may be conducted analytically using computer models to define a tradespace. The development of a tradespace and its exploration is predicated on the idea that a design problem can be expressed, at least in part, mathematically. This may be used, in combination with MBSE, to link architectural

products with external models and simulations (Beery 2016) to "illuminate the tradespace" (Paulo, Beery, and MacCalman forthcoming).

1. Tradespace Usage in the Literature and Definition

The term "tradespace" is widely used in the literature, but rarely rigorously defined. Brantley, McFadden, and Davis (2002), Ross and Hastings (2005), Sitterle et al. (2015), the Pennsylvania State University Applied Research Laboratory (2015) and Buede (2000) provide varying definitions:

> The "trade space" can be defined as the set of program and system parameters, attributes, and characteristics required to satisfy performance standards. Decision makers define and refine the developing system by making tradeoffs with regard to cost, schedule, risk, and performance; all of which fall within the systems trade space. (Brantley, McFadden, and Davis 2002, 2)

> **Tradespace.** Is the space spanned by the completely enumerated design variables, which means given a set of design variables, the tradespace is the space of possible design options. … Using models and simulation, the full set of design options—the tradespace—can be evaluated in terms of benefits and costs to decision makers. Often the Utility-Cost plot will be referred to as the tradespace as well since it is a useful representation for making "best" system value trade decisions. … The Pareto Front is the tradeoff curve between metrics. (Ross and Hastings 2005, 2)

> A tradespace is defined as a collection of design variables and system attributes, different levels of which characterize each design alternative for a given system. A model or collection of models acts as a mathematical representation of the system, often with external variables to map the input variables to output variables. Commonly, input variables are chosen to be system design variables while output variables are defined to be system attributes. This relationship may be reversed depending on the mapping, and the delineation between which design variables are used as inputs and which are derived via model transfer functions is not always clear. Variables may be intrinsic to the system or dependent on conditions external to the system (e.g., cargo space versus miles per gallon). Some form of cost is also typically derived from the characteristics that describe each system design alternative. (Sitterle et al. 2015, 651)

> 1) It is a ***shopping process***. The decision maker discovers what it is they want while they are looking for it.

2) It is a ***negotiated process***. Decisions of real complexity involve multiple decision makers, each with their own motives and levels of expertise.

3) It is an ***iterative process***. The trade space is first *explored*, and then the knowledge gained is *exploited* by focusing future searches to regions of decreasing breadth but of increasing depth and detail.

(PSU-ARL 2015)

Buede (2000) does not explicitly use the term tradespace, but he provides a visual depiction of the tradespace as seen in Figure 2. Notably, he indicates that there is a back-and-forth (indicated by two-way arrows) of different requirements and objectives along with cost and performance trade-offs that all, together, inform the tradespace.

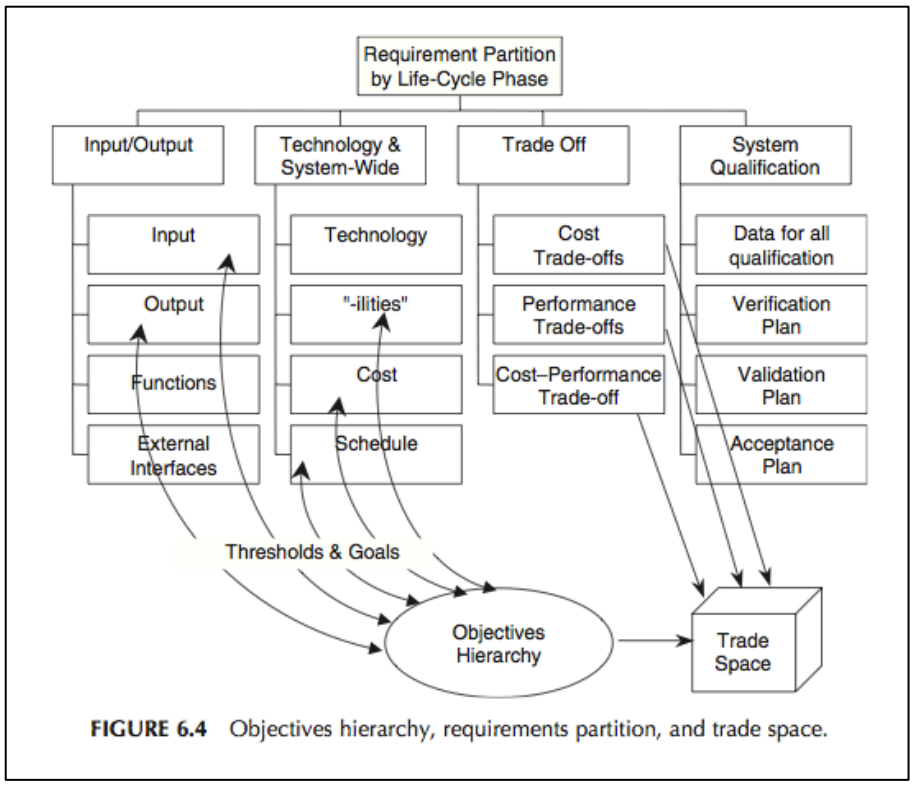


Figure 2. Buede Tradespace and Design Problem Definition through Requirements. Source: Buede (2000)

Collectively, these definitions share a few key aspects. The first is that there must be a manner to assess all feasible design points. Feasible must be defined in a practical as

opposed to theoretical sense and account for the various constraints that affect what may make a design point possible (e.g., the time to develop the system affects what technology may feasibly be considered. A system to be implemented within a year can only consider computing power on order of contemporary computing power. A system to be implemented in 10 years may account for Moore's Law). The second shared aspect of a tradespace is that, for each design point, there must be an associated set of system attributes—cost, performance, and other key factors. This may be done as an enumerated list, or, more generally, a function that takes design parameters as an input and outputs system attributes. Finally, there is an associated set of requirements and constraints that define what is and is not desirable with regard to system attributes. Combining these three leads to a set of potential design points that may be considered the tradespace. These concepts may be expressed more rigorously mathematically.

2. Mathematical System Design and Tradespace Definition

A system design problem may be defined in a general, abstract manner. Wymore (1993) developed his tricotyledon theory to characterize what he called the functionality, buildability, and implementability cotyledons. These are sets of theoretic system design points that, respectively, meet system operational requirements, feasibility requirements, and their intersection, as depicted in Figure 3. Wymore's language and description are, unfortunately, outdated and esoteric. Analogously, Statnikov and Matusov (2002) present their "Parameter Space Investigation" that uses more common set theoretic and mathematical optimization language to describe design problems. A sample depiction of their work in two-dimension is seen in Figure 4. In general, one can describe a system design problem mathematically by defining design parameters, design points, attribute functions, and utility functions. System design points are defined according to a set of parameters. System attributes are defined by the attribute functions that take design parameters as an input and output an attribute value. Utility functions prescribe a normalized value for each attribute. This, combined with a relative weighting of attributes may form an optimization problem in which the designer may assess the best design.

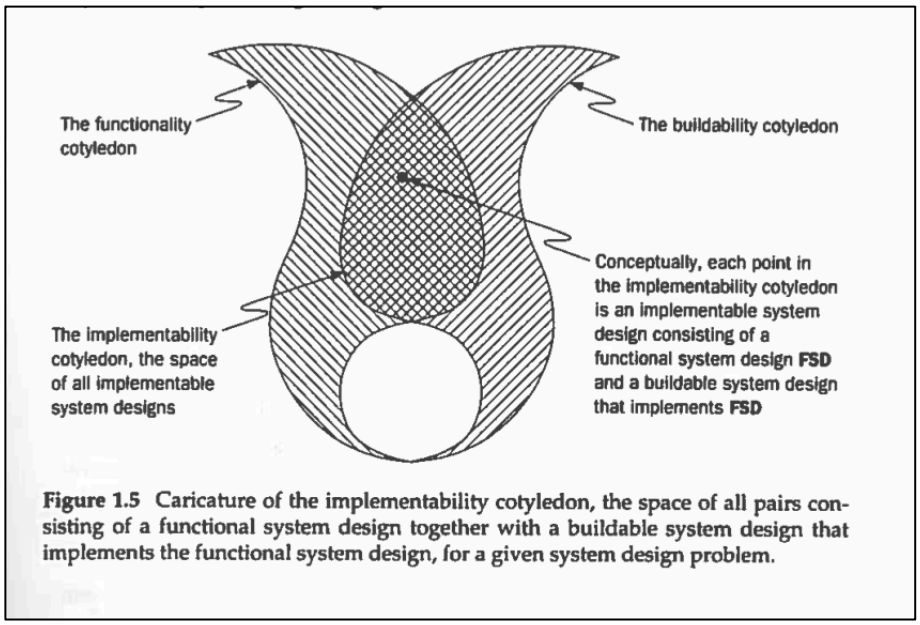


Figure 1.5 Caricature of the implementability cotyledon, the space of all pairs consisting of a functional system design together with a buildable system design that implements the functional system design, for a given system design problem.

This caricature depicts the three theoretical spaces a systems engineer must contend with in engineering a system: The functional, buildable, and implementable cotyledons.

Figure 3. Tricotyledon Theory. Source: Wymore (1993)

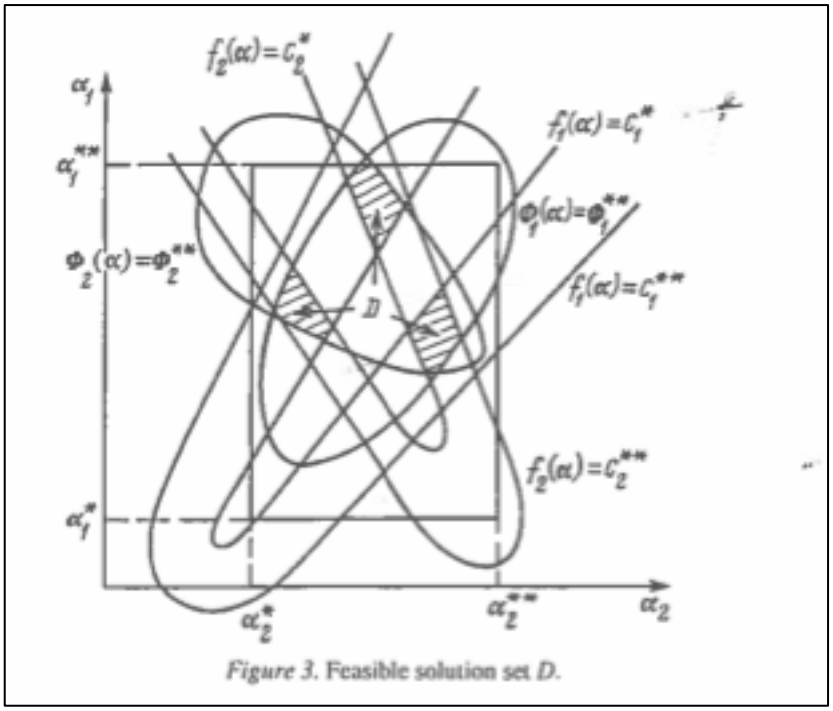


Figure 3. Feasible solution set *D*.

Figure 4. Parameter Space Investigation Example. Source: Statnikov and Matusov (2002)

For this dissertation, the general mathematical formalization of a design problem is defined in the following sections.

a. Design Point and Design Space

A system **design point** may be described according to its various **parameters**. Call the i^{th} design point:

$$\boldsymbol{d_i} = < d_{i1}, d_{i2}, \ldots d_{ij}, \ldots\ d_{ik} >$$

The design point has k parameters, $d_{ij}, 1 \leq j \leq k$. Each parameter is defined on its domain, a closed set $\boldsymbol{D_j}$. Example parameters include:

- Engine Type, which may be defined on the set $< diesel, gasoline, electric >$
- Car Color, which may be defined on a set such as $< red, blue, gray, green >$ or a set $< [r, g, b] | \, r, g, b \in \mathbb{Z}; 0 \leq r \leq 255, 0 \leq g \leq 255, 0 \leq b \leq 255 >$ (the RGB color model https://en.wikipedia.org/wiki/RGB_color_model).
- Car length, which may be on $[0, 5] \subset \mathbb{R}$, the number of meters the car may be long.

The **design space** is the set of all possible design points. It is the Cartesian product of all of the parameter domains. Call the design space:

$$\boldsymbol{D = D_1 \times D_2 \times \ldots \times D_j \times \ldots \times D_k}.$$

If $\left|\boldsymbol{D_j}\right| \leq m \in \mathbb{R}$ for some positive m, then $|\boldsymbol{D}|$ is finite (if large), otherwise, the design space is infinite, but still closed. Note that if $\boldsymbol{D}$ is infinite and not countable (say some $\boldsymbol{D_j}$ is a subset of $\mathbb{R}$), then one cannot enumerate the $\boldsymbol{d_i}$. For this dissertation, we assume a discrete, finite definition of each $\boldsymbol{D_j}$, thus the design space is finite, if large.[4] In cases where parameters are defined on a continuous domain, we may approximate them by choosing a number of discrete levels representative of the domain. For example, the length of a vehicle may have a domain of 1 to 10 meters; this may be approximated as the domain: <1m, 2m, 3m, 4m, 5m, 6m, 7m, 8m, 9m, 10m>.

[4] We may assume this as each potential constituent system is a discrete element. Each operational activity or rule of employment is similarly a singular, discrete element. Each organization is a discrete element.

b. Environment

As every system exists in a larger context, there are environmental parameters that may affect how a design point performs in that situation. This environment may be physical (e.g., terrain or weather), regulatory (e.g., interface standards or government rules and regulations), or the behaviors of external actors (e.g., enemy activity). The important distinction of an environmental parameter from a design parameter is that the designer has no control over environmental parameters and does have some level of control over design parameters. An environmental point may be described as a vector of environmental parameters:

$$\boldsymbol{e_m} = < e_{m1}, e_{m2}, \dots, e_{mk} >$$

Where each e_{m*} describes the relevant parameter. The set of all possible environmental parameters is $\boldsymbol{E}$.

c. System Attributes

Each design point has some set of **system attributes** that are defined by a function. If there are *x* attributes, these functions may be termed:

$$f_a: \boldsymbol{D} \rightarrow \boldsymbol{R_a}, 1 \leq a \leq x$$

with

$$\delta_{ai} = f_a(\boldsymbol{d_i})$$

where $\boldsymbol{R_a}$ is a closed set, commonly a subset of $\mathbb{R}^n$. The set of all attributes for design point $\boldsymbol{d_i}$ is $\boldsymbol{\delta_i}$. The set of all functions is $\boldsymbol{f}$.

Common examples of systems attributes include:

- The cost of a design point.
- The mean time between failures of a design point.
- The operational performance of a design point.
- The availability of a design point.
- The feasibility of a design point.

Note that each f_a must be well defined. This definition may be analytic (e.g., the COSYSMO model for cost) or through the results of a simulation or a meta-model developed based upon the results of selected design points and subsequent statistical inference.

Thus far, we have assumed that the environment is static; however, this is not always true. In this case, the attribute function f_a may be modified to include environmental parameters. That is, one may say:

$$\delta_{aim} = f_a(\boldsymbol{d_i}, \boldsymbol{e_m})$$

Is the a^{th} attribute of the i^{th} design point in the m^{th} environment. In a more detailed analysis, this may be a further useful consideration as a decision maker must vary what potential environments in which a system must operate. All subsequent discussion assumes that the environment is static.

d. Acceptable Design Points

With this framework in place, a designer may place acceptable boundaries on the design space and the system attributes based upon criteria (these may be engineering, political, or of another nature) of the designer's choosing. For each $\boldsymbol{D_j}$ there is some D_j^{min} and D_j^{max}. Similarly, for each δ_{a*}there is an associated δ_{a*}^{min} and δ_{a*}^{max}. Together these serve to constrain the set of allowable design points, call this subspace, the set of **acceptable design points**, $\boldsymbol{D^A} \subset \boldsymbol{D}$,

$$\boldsymbol{D^A} =< \boldsymbol{d_i} \in \boldsymbol{D} | \forall d_{ij} \in \mathbf{d_i}, D_j^{min} \leq d_{ij} \leq D_j^{max} \; and \; \delta_{a*}^{min} \leq \delta_{ai} = f_a(\boldsymbol{d_i}) \leq \delta_{a*}^{max}, \forall f_a \in \boldsymbol{f} >$$

Once a designer has defined $\boldsymbol{D^A}$, if it is non-empty, the question, of course, is what is the best choice of design point?

e. Choosing a Design Point

The term "best" depends significantly upon the values a decision-maker assigns to each system attribute and the relative weighting among those functions. For each attribute, assign a utility function, $u_a: \boldsymbol{R_a} \rightarrow [0,1], 1 \leq a \leq x$ that describes the value the

decision-maker assesses for that attribute. These utility functions may take many forms, examples include:

- An S curve, indicating initially low returns, followed by rapidly increasing returns, and then decreasing returns.
- An inverse logarithmic curve, indicating decreasing returns.
- An inverse parabola, indicating the desire for a value in the middle of $\boldsymbol{R_a}$.

For each $u_a(\delta_{a*})$ the decision-maker may further assign a minimum utility, $\mu_a \in [0,1]$. In most cases, it makes sense to assign $\mu_a = 0$ and assess a minimum for the attribute according to $\boldsymbol{D^A}$.

The decision-maker further assigns a relative weight to each attribute, $w_a, 1 \leq a \leq x, with\ \sum_1^x w_a = 1$. This leads to an optimization problem:

Maximize: $w_1 \cdot u_1\big(f_1(\boldsymbol{d_i})\big) + w_2 \cdot u_2\big(f_2(\boldsymbol{d_i})\big) + \cdots + w_x \cdot u_x\big(f_x(\boldsymbol{d_i})\big)$

subject to

$$\boldsymbol{d_i} \in \boldsymbol{D^A}$$

$$u_a\big(f_a(\boldsymbol{d_i})\big) \geq \mu_a$$

If all of the above functions are well defined and $\boldsymbol{D^A} \neq \emptyset$, this problem may be solved, or closely approximated, using mathematical programming. Call the results, the set of optimal points, $\boldsymbol{D^{A*}}$

An alternative to optimization is satisfaction. In this manner, a decision-maker merely defines $\boldsymbol{D^A}$ and states that any point in $\boldsymbol{D^A}$ is satisfactory.[5] This may be useful in cases where optimization is difficult, such as when u_a is unknown or poorly known. One may further consider the set of Pareto optimal points, $\boldsymbol{D^{AP}} \subset \boldsymbol{D^A}$. These are defined as:

$$\boldsymbol{D^{AP}} =< \boldsymbol{d_i} \in \boldsymbol{D^A} \mid \nexists \boldsymbol{d} \in \boldsymbol{D^A}\ f_a(\boldsymbol{d_i}) \geq f_a(\boldsymbol{d}), 1 \leq a \leq x\ and\ f_a(\boldsymbol{d_i}) > f(\boldsymbol{d}) >.$$

Stated simply, a point is Pareto optimal if one cannot improve one attribute without worsening another. Note that $\boldsymbol{D^{A*}} \subseteq \boldsymbol{D^{AP}} \subseteq \boldsymbol{D^A}$.

[5] In reality, the most common application is that a designer defines and evaluates several options and then chooses the best among them, where best is defined based upon the decision-maker's values.

f. Implications of This Formalization

The most obvious implication stems from the fact that, $\boldsymbol{D^{A*} \subseteq D^{A}}$**.** If one further restricts any or all of the $\boldsymbol{D^{A}}$ by making $\tilde{D}_j^{min} > D_j^{min}$ *or* $\tilde{D}_j^{max} < D_j^{max}$, or similarly by making a $\tilde{\delta}_{a*}^{min} > \delta_{a*}^{min}$ *or* $\tilde{\delta}_{a*}^{max} < \delta_{a*}^{max}$, there is a new $\boldsymbol{\tilde{D}^{A} \subseteq D^{A}}$. The set of optimal solutions on $\boldsymbol{\tilde{D}^{A}, is\ \tilde{D}^{A*} \subseteq D^{A*}}$**.** Accordingly, as one restricts $\boldsymbol{D^{A}}$, the possible set of optimal solutions is further restricted. Similarly**,** if one defines two disjoint sets of acceptable solutions, $\boldsymbol{D^{A}}$ *and* $\boldsymbol{\overline{D}^{A}}$**,** then $\boldsymbol{D^{A*}}$ *and* $\boldsymbol{\overline{D}^{A*}}$ are disjoint. Furthermore, if one varies $f_a, u_a,$ or w_a, the solution to the optimization problem is similarly changed. The choice of the best design point, then, heavily depends upon the limitations placed upon the design parameters and the system attributes and the utility assigned to each parameter and its relative weight.

In an ideal world, $f_a, u_a,$ and w_a are defined *a priori*, the limits that define $\boldsymbol{D^{A}}$ are also pre-defined and the most significant challenge is in defining $\boldsymbol{D^{A}}$, its associated attributes, and then interrogating the space. This is not an insignificant challenge. In some cases, the spaces are huge, and one must carefully select design points for analysis (by which to define the attributes) and, potentially to define an approximation to any f_a. More problematic than defining the attribute functions, however, is that the limitations placed upon the design space and the utility functions may be somewhat arbitrary—subject to personal whims, pre-conceptions, or other factors. So, while one may conduct an optimization, and, if the set of allowable design points is not empty, one will get at least one optimal point, the reality is, that this may not truly satisfy the stakeholders. For this reason, the concept of the tradespace was born.

g. Mathematical Definition of Tradespace

For this dissertation, a tradespace is defined based upon the aforementioned aspects. A tradespace is the set of potential design points ($\boldsymbol{D}$), their associated attributes ($\boldsymbol{\delta_i}$), and the bounding requirements ($D_j^{min}, D_j^{max}, \delta_{a*}^{min}, and\ \delta_{a*}^{max}$) that together define the sub-set of acceptable design points ($\boldsymbol{D^{A}}$) from which an engineer, analyst, or

decision-maker may choose a system design by any number of means—optimization of utility, heuristic selection among Pareto optimal points, or some other method.

h. Conclusion

It is a non-trivial problem to define the tradespace for a system of even moderate complexity. Further, to be useful, a tradespace must be linked to standard architectural products. Accordingly, researchers have defined various methodologies for using MBSE in conjunction with tradespace exploration.

C. MODEL-BASED SYSTEMS ENGINEERING

INCOSE defines MBSE as: "the formalized application of modeling to support system requirements, design, analysis, verification and validation, beginning in the conceptual design phase and continuing throughout development and later life cycle phases" (Friedenthal et al. 2007, 5). More concretely, the central tenant of MBSE is that systems engineers move from a "document centric" to a "model centric" approach (Friedenthal et al. 2007, 4). The purpose of this is to, "enhance[s] the ability to capture, analyze, share, and manage the information" (Friedenthal et al. 2007, 7). This realizes five principal benefits

1. "Improved communications."
2. "Increased ability to manage system complexity."
3. "Improved product quality."
4. "Enhanced knowledge capture."
5. "Improved ability to teach and learn systems engineering fundamentals." (Friedenthal et al. 2007, 7)

The INCOSE definition of MBSE modifies the phrase "application of modeling"[6] with the word "formalized." This is the essence of MBSE, methodologies and tools that link the different aspects of systems engineering. So, while systems engineer have always used models, these disparate models have not been formally linked in such a manner that

[6] The author assumes the reader is familiar with modeling and simulation. For a greater treatment, see Law (2008) or Sokolowski and Banks (2011) among others.

a change in one propagates changes in the others. This is the utility of MBSE—such linkages facilitate the above-mentioned benefits.

MBSE is conducted through the use of modeling languages, methods and tools. Estefan (2007) provides a useful overview of various MBSE methodologies, tools, and languages. It has been used to solve a wide variety of problems across various disciplines. For the DOD, examples of MBSE application include engineering for Space Systems (Jepperson 2013), Supply Chain Management (Bonagrazia-Healy et al. 2014), Energy Efficiency in a Marine Operational Setting (Bennett et al. 2014), Naval Ship Design and Mine Warfare (Pisani, 2013; Frank et al., 2014; Kaymal 2013).

While MBSE is generally applicable to systems engineering at large, most MBSE research has focused on various aspects of systems architecting (Beery 2016). Increasingly, recent research has advanced the state-of-the-art (e.g., Beery 2016) to include greater aspects of systems engineering (i.e., analysis) in conjunction with architecting. This is commonly expressed, at its end state, through a tradespace. While this end state is useful, the methodologies and tools to define this tradespace are of greater importance.

1. Model-Based Systems Engineering for Design

Until recently, there was a significant gap in the MBSE state-of-the-art. The majority of MBSE research occurs in the area of systems architecting (Beery 2015). This has created an artificial separation between systems architecting and systems analysis (Beery 2015) as seen in Figure 5. This is problematic, as, "that research has focused primarily on development of system architecture models and has largely ignored the need to clearly link systems architecture models to detailed external models and simulations" (Beery 2016, 3). To address this limitation, Beery (2015) developed the MBSE Methodology for Employing Architecture in Systems Analysis (MEASA).

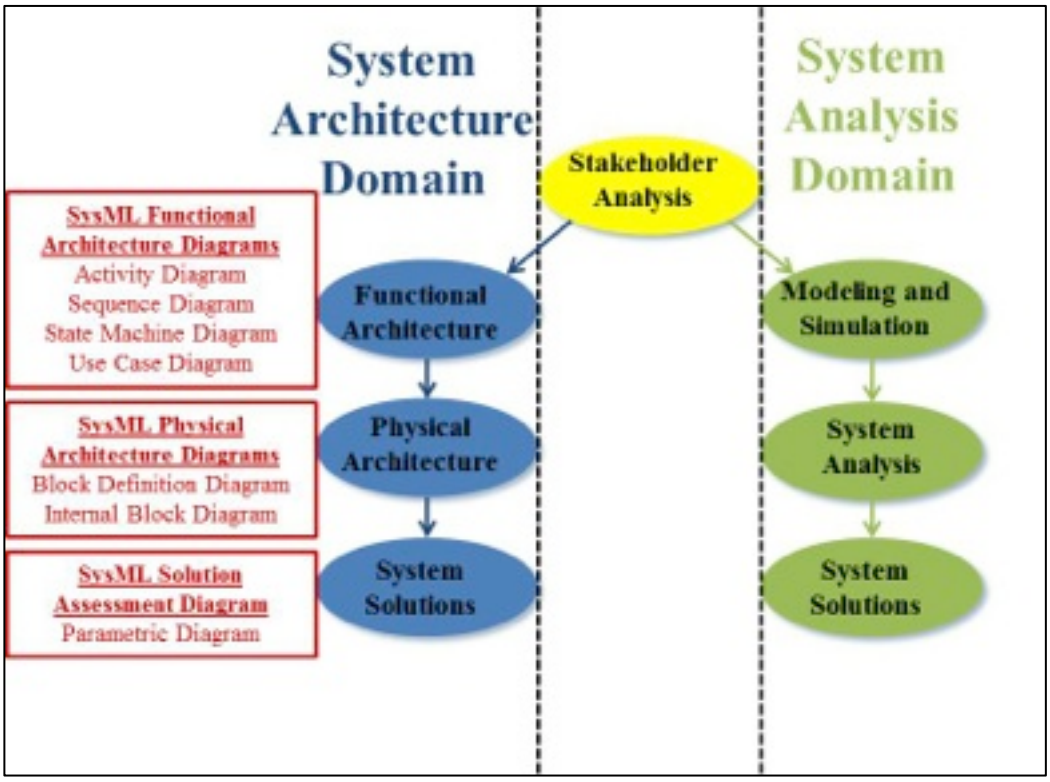
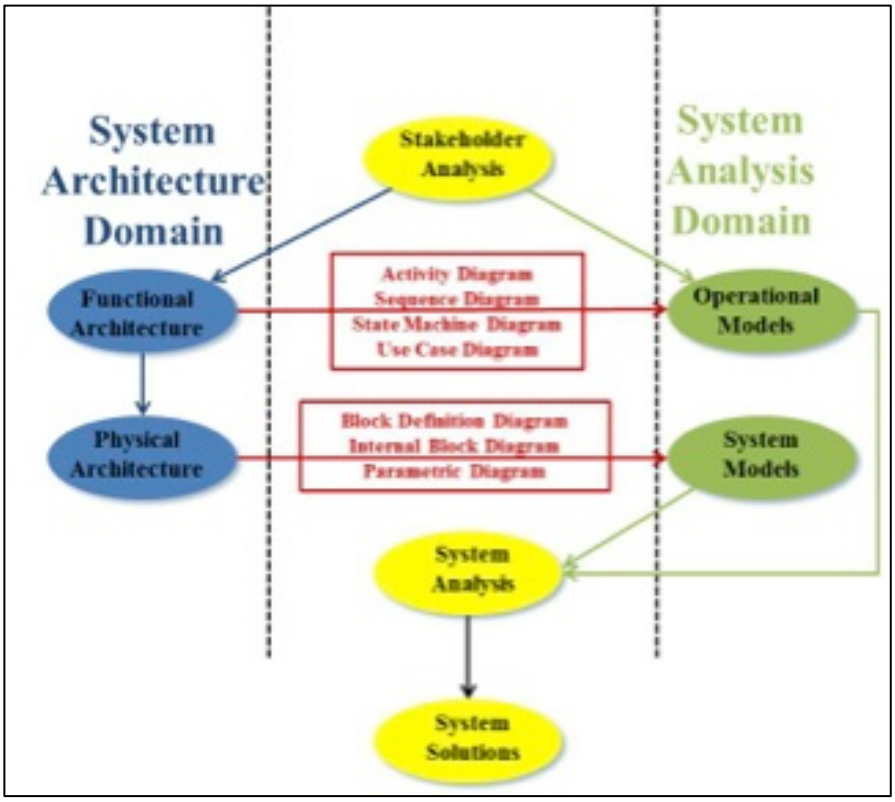


Figure 5. Beery Depiction of Current MBSE Research Focus.
Source: Beery (2015)

a. Model-Based Systems Engineering Analysis Methodology Description

Beery's (2015) MBSE MEASA methodology links two systems engineering domains, architecture and analysis, as depicted in Figure 6. This facilitates exploratory design as one can use this methodology to define a tradespace.



This figure depicts how Beery's MBSE Analysis Methodology can be used to link Systems Architecting with Systems Analysis to improve early life cycle system design.

Figure 6. Beery's MBSE Analysis Methodology Utility.
Source: Beery (2015)

The intent of the MBSE MEASA is "to be utilized for definition, design, and analysis of large scale, complex systems early in the system design cycle" (Beery 2016, 56). It is not applicable to systems integration or implementation. Furthermore, MEASA assumes that a valid systems engineering problem and need have been identified in accordance with typical systems engineering methods (Beery 2016). Finally, Beery intends MEASA to be nested within the greater context of MBSE, e.g., the use of SysML (Beery 2016).

The MEASA is intended to support the development of systems engineering artifacts typically associated with problem definition, system design, and system analysis as identified by systems engineering textbooks such as (Blanchard and Fabrycky 2010; Buede 2000) and articulated by Beery (2016). As MEASA supports the development of

these artifacts, it can be used in conjunction with any specific systems engineering methodology (e.g., the waterfall, vee, or spiral) (Beery 2016).

The MEASA is depicted in Figure 7. In it, one sees how the methodology links the two domains of systems architecting and analysis. The left hand side of the figure depicts systems analysis, which involves modeling how the system performs in an operational environment. The right hand side of the figure depicts (high -level) systems architecture through a system synthesis model. The center shows how the two are linked in MEASA. In total, this figure captures Beery's MEASA, and provides an overview for how a researcher or engineer may employ MBSE to link systems architecting with systems analysis during early life cycle system design.

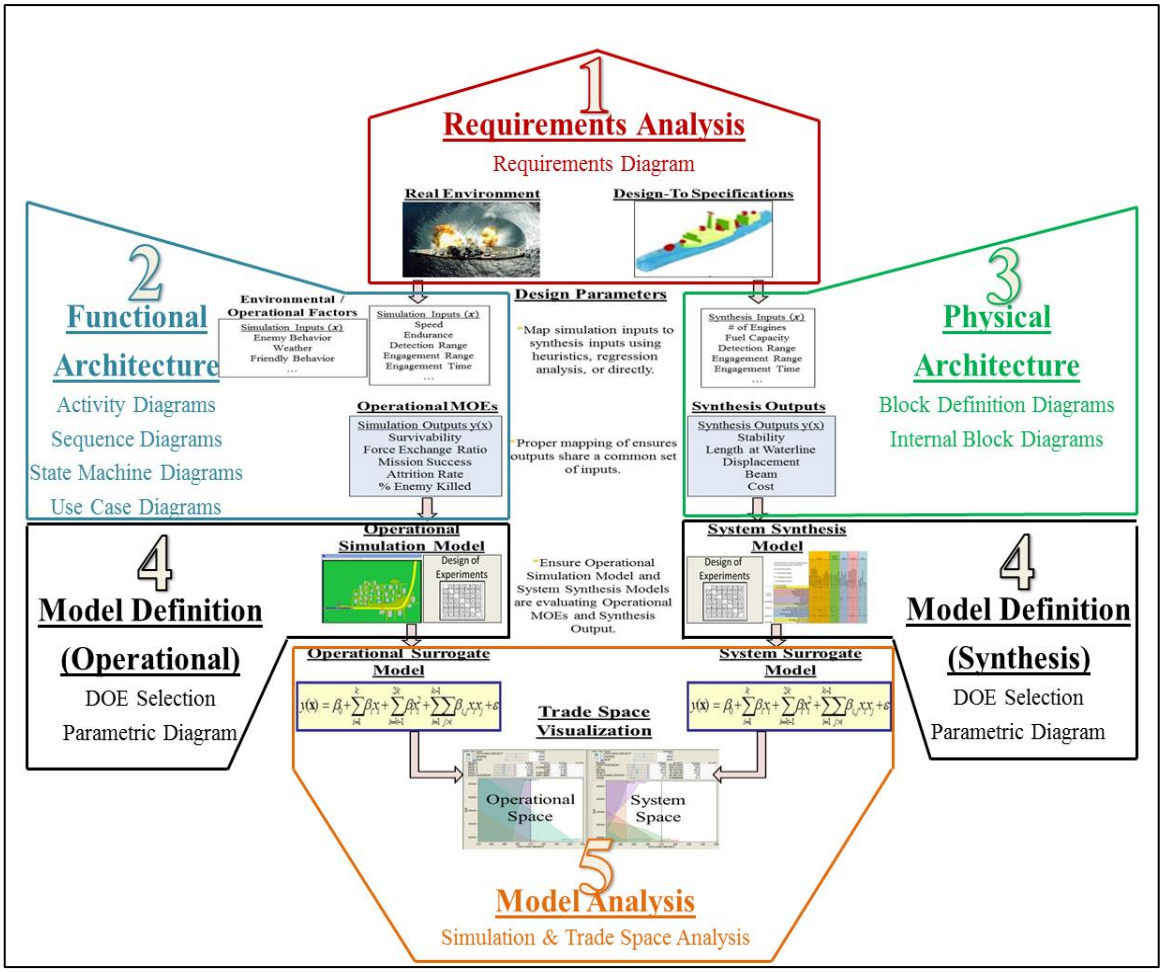


Figure 7. Beery's MBSE MEASA. Source: Beery (2016)

The MEASA begins with the development of operational simulation models (Beery 2016) and is depicted by approximately the left half of Figure 7. In this, the system is modeled functionally and operationally against a range of operational and environmental variables. It is then assessed against the refined need(s) defined during the problem definition phase (Beery 2016). Importantly, during this phase, statistically relevant variables are identified using standard statistical analyses such as analysis of variance (ANOVA) or other appropriate methods (Beery 2016). These are called design parameters in Figure 7. Once relevant parameters are identified, an engineer can conduct a DOE, assess the design points and develop a surrogate model of operational performance that takes environmental and design parameters as inputs and outputs operational MOE (Beery 2016).

The second major step of MEASA is the development and analysis of the system synthesis model(s) (Beery 2016). A system synthesis model is one which takes system design parameters as inputs and outputs both the feasibility of a design with such parameters (i.e., an assessment that says a system with such parameters may be built given the set of constraints) and the projected system characteristics (e.g., cost or weight). The creation of such a synthesis model is, tacitly, a high-level systems architecture. In Beery's example, the system synthesis model, the architecture is that of a ship, and there is a model used by naval architects that relates the number of engines, ship length, crew, and so forth to determine if the ship is feasible, and what its cost, stability, and other characteristics are (Beery 2016).

The final major step of MEASA is linking the previous two steps (Beery 2016). This is the particularly innovative step, in which Beery developed the MEASA to formally develop a method to link systems analysis (step one) with systems architecting (step two). In Figure 7, the two boxes labeled design parameters show the input parameters to both the operational and synthesis models. Beery develops an explicit linkage between these variables. In some cases, there is a very obvious one-to-one correlation, such as the number of helicopters as a synthesis parameter and an operational parameter. In other cases, there is a more complex relationship, for example, the simulation input of ship range may be dependent upon both the number of engines and

the fuel capacity according to some formula (Beery 2016). This linkage and the previous modeling efforts are displayed through the use of a dynamic "dashboard" as indicated by the tradespace in Figure 7. This tradespace is an example of an exploratory design decision-making methodology, as previously described.

b. MBSE MEASA Limitations

The MEASA, as developed, is applicable to developing material system solutions and monolithic systems (as opposed to SoS) (Beery 2016). The reason for this is because the MEASA assumes that 1) There is a feasibility (synthesis) model for the system in question, 2) One may define a set of operational parameters for the use in operational simulations. These operational parameters may be defined through functions that take design parameters as input and output these operational parameters, and 3) Attribute functions—synthesis or operational—may generally be defined through the use of DOE and meta-models. These assumptions are not generally true in the case of SoS; particularly if one wishes to represent an SoS completely by including process and organizational parameters.

The first limitation of the MEASA to SoS is that one must have a system feasibility model to assess if a given design point is feasible. For example, in the application of the MEASA, Beery (2016) demonstrates how the design parameters for a ship are related; e.g., the length of the ship directly affects the number of helicopters that may be employed due to space requirements. This assumption is reasonable for systems whose feasibility is a function of physical parameters—there are well known physical models for a large variety of domains. When one begins to consider organizational and process parameters, however, this situation is less well defined.

The second limitation of the MEASA to SoS is that it defines a system design problem in a somewhat unique manner from the description in the Section II.B.2. In this dissertation, there are only design points, $\boldsymbol{d}$, and attribute functions, f_a. These attributes may be of any type e.g., operational performance, cost, feasibility. Beery (2016) defines two distinct sets of parameters—design and operational. Call the design parameters $\boldsymbol{d} \in \boldsymbol{D}$ as usual, and call the operational parameters:

$$\boldsymbol{o} \in \boldsymbol{O} =< \{o_1, o_2, \ldots o_p\} | o_i \; is \; an \; independent \; operational \; parameter >$$

Furthermore, there are a distinct set of system attribute functions that take operational attribute points as input and output operational measures of performance, call these $g_b: \boldsymbol{O} \rightarrow \boldsymbol{R_b}$. These are the operational corollaries of $f_a: \boldsymbol{D} \rightarrow \boldsymbol{R_a}$. Figure 8 clarifies this to demonstrate the MEASA in this dissertation's mathematical notation.

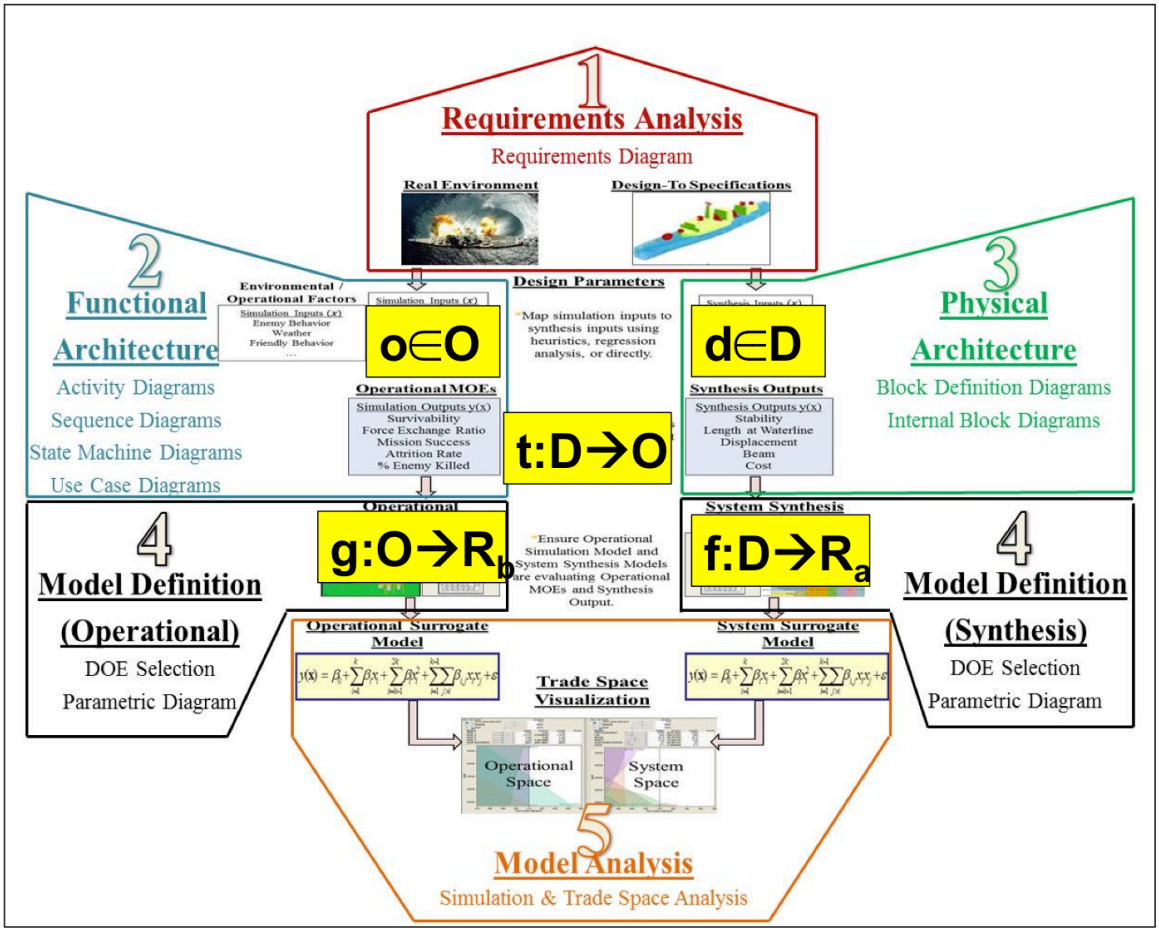


Figure 8. Overlay of Current Work's Notation on the MEASA. Adapted from Beery (2016).

The reason for partitioning design and operational variables is practical; operational models typically require input variables that are operational in nature (e.g., an agent based model considers vehicle speed as a variable, not number of engines, vehicle

weight). This is acceptable, but to do this, one must have a well-defined method of defining the function that links these two sets of parameters, a transfer function:

$$t: \boldsymbol{D} \rightarrow \boldsymbol{O}$$

In a physical model, this is often well understood. For example, one may define a transfer function that takes the system weight, shape, and engine size as inputs and outputs speed. In doing this, one may consider the problem $g\big(t(f(\boldsymbol{d}))\big)$ to define the operational system attributes of a system design point. Alternatively, one may consider the problem $f\big(t^{-1}(\boldsymbol{o})\big)$[7] to define the synthesis system attributes of an operational set of parameters. If the design space is limited to physical parameters, it is reasonable to assume that one may define such a transfer function—there are well-understood relationships among physical design parameters and performance in many cases as demonstrated by Beery (2016). In cases in which this transfer function is poorly understood, the alternative is to only define system attributes via design parameters.

The final challenge of Beery's (2016) MEASA is that it makes extensive use experimental design and meta-models to define attribute functions. DOE for problems with qualitative variables are best when those variables are limited to 10 or fewer levels (Sanchez and Wan 2012). For SoS, this is problematic as one may quickly exceed this threshold as, for the set of SoS that may be formed from *n* potential systems forms a qualitative variable with approximately 2^n levels. Furthermore, while there are a number of DOE that address 2nd order interactions (Vieira et al. 2011; MacCalman 2013), an SoS with necessarily involves higher order interactions that are statistically significant, especially among its categorical variables (i.e., ones defined against physical, organizational, and process parameters), these options are impractical. DOE for 3rd order interactions are an area of active research; 4th and higher are beyond the state-of-the-art (Kleijnen et al., 2005).

[7] Note: $t: \boldsymbol{D} \rightarrow \boldsymbol{O}$ is well defined. That is, for a given system design, one will only get a single operational parameter (e.g., a design won't give two different maximum speeds), although two designs may yield the same operational parameters. On the other hand, $t^{-1}: \boldsymbol{O} \rightarrow \boldsymbol{D}$ may not be well defined. That is, an operational parameter may be achievable by multiple system designs.

Combined, these limitations make the MBSE MEASA ineffective for application to SoS, particularly SoS described by their full physical, process, and organizational perspectives. Beery specifically notes this in his areas of future research section (Beery 2016).

2. Conclusion

MBSE is the desired future state of the practice per INCOSE's strategic vision (INCOSE 2015). The transformation from document-based systems engineering to MBSE is an ongoing process and has been made possible by the large variety of research in MBSE tools, methods, and applications. Beery's MEASA is an important advancement in the state-of-the-art, particularly as it rigorously links two key areas of systems engineering, architecting and analysis. This facilitates subsequent tradespace development and TSE and improves design decision-making.

As with any new methodology, a test of its utility is to apply it broadly. Beery (2016) demonstrated the MEASA in the context of a relatively well-defined problem for a monolithic system. SoS are a somewhat more complex and distinct subset of systems engineering with unique challenges and approaches for solving these challenges. There is, therefore, a significant utility in addressing the shortfalls of the MEASA as applied to SoS. The subsequent section discusses SoS, SoS engineering, and their relationship with MBSE and the MEASA.

D. SYSTEMS OF SYSTEMS ENGINEERING

SoS are a significant subfield of systems engineering. SoS, while being systems in their own right, have unique characteristics that warrant unique engineering approaches across the spectrum of systems engineering, including problem definition, architecting, analysis, integration, implementation, and management. Multiple researchers and practitioners have developed various methods and tools to contend with these distinct challenges (Maier 1998; DOD 2008; Jamshidi 2008; Jamshidi 2009; Rainey and Tolk 2015). This section defines SoS, the implications of SoS for systems engineering, and

outlines the current methods of SoSE. It further places this research in the greater context of SoSE.

1. Systems of Systems

a. SoS Definition

Maier (1998) laid some of the foundational work for SoS. In it, he defined an SoS as a group of distinct systems characterized by operational and managerial independence, exhibiting emergent behavior, geographically dispersed, and evolutionary in their development (Maier 1998). This definition and classification has been widely adopted and expanded upon with additional characteristics such as autonomy, belonging, connectivity, diversity, self-organization, and adaptation (Boardman and Sauser 2006; Sage and Biemer 2007). The DOD (2008, 4) defines SoS similarly: "A SoS is defined as a set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities." Regardless of the precise definition, the general concept is that an SoS is a system, composed of multiple independent systems, that provide some capability, and that the total design or operation of the system is not wholly controlled by any one entity. As Maier's (1998) definition is so common in the literature, his characteristics are outlined as follows:

(1) Operational Independence

Each constituent system is a purposeful, useful system in its own right. It can operate in its intended environment and accomplish a mission (Maier 1998). For example, a patriot missile battery is an independent air defense system that can conduct air defense operations on its own; it is also a member of a more general missile defense SoS. A counter-example is the engine of an aircraft; it is, in many senses, a system in and of itself, but it is not operational or useful without the rest of the aircraft, therefore it is a sub-system vice a constituent system.

(2) Managerial Independence

The constituent systems are managed by independent entities. This implies that each constituent has its own life cycle, maintenance and upgrade criteria, and is generally run by its own program (Maier 1998). An example of this is two distinct defense programs of record. Though they may both support a common goal, each system is managed independently and run by its own program executive officer (PEO). A counter-example is two sub-systems within a single program of record. Though independent design teams may be working on each sub-system, final decisions about their design rest with the program manager.

(3) Geographic Dispersion

The constituent systems of an SoS are generally geographically dispersed. The actual distances involved are relative; an SoS may be dispersed by meters, kilometers, or hundreds of kilometers. Importantly, as a result, the constituent systems do not generally exchange material or energy; rather, the primary interface among the various constituent systems is information (Maier 1998). An example of this is a kill chain in which various constituent systems conduct different steps of the kill chain and pass on the information of what has been conducted and the target's location and status.

(4) Evolutionary Development

SoS are evolutionary in nature. This is a direct result of the managerial and operational independence of the constituent systems. As the constituent systems are thinking, adapting, and reacting independent actors capable of making decisions, the SoS will necessarily evolve with their changing behavior (Maier 1998). Moreover, as each constituent system exists on its own life-cycle, constituent systems will retire from and be introduced into the SoS at different times. The SoS must evolve to adapt to these changes.

(5) Emergent Behavior

SoS exhibit emergent behavior. This is defined as a behavior that is not entirely contained by any constituent system (Maier 1998). Emergence occurs at various levels:

simple, weak, strong, and spooky (Maier 2015). These levels are differentiated by our ability to understand, predict, and model the behavior. Emergence in an SoS is both a desirable behavior (for the desired SoS capabilities) and an undesired behavior (unpredicted, negative behavior). Ultimately, the goal of SoSE is to design an SoS that produces desired emergent behaviors and minimizes non-desired ones; Maier (2015) states, "To be an SoS, the collective must possess properties or behaviors that are not possessed by any of the components. This is an 'emergent property.'" The four categories of emergence are defined as follows:

- Simple: Emergence that is readily predicted through an understanding of the constituent systems and readily modeled (Maier 2015)
- Weak: Emergence that is replicable with a simulation and may be understood after it is recognized. An example of this would be traffic patterns on a communications network (Maier 2015).
- Strong: Emergence that is either not replicable or highly difficult to replicate in a model or simulation, but is consistent with the known properties of the constituent systems. An example of strong emergence would be the human brain. We cannot replicate its function in a model, but it is entirely consistent with current understanding of neurons (Maier 2015).
- Spooky: Emergent properties that are not replicable in a model and are inconsistent with the known properties of the constituent systems. There are no known examples of this sort of emergence (Maier 2015).

With these definitions in mind, one can see that is only truly possible to design an SoS that exhibits simple or weak emergence as these may be modeled and behaviors may be replicated and predicted. Strong emergence may be a factor in an SoS, but only in an evolutionary and reactionary manner. Spooky emergence has no obvious examples and cannot be designed by definition. This research is focused solely on SoS design; accordingly, only simple or weak emergent properties are considered.

b. Delineation between Systems and Systems of Systems

There is no strict delineation between systems and SoS. Rather, the identification of a system as a singular system versus an SoS allows engineers to tailor their approach in the manner that is most useful for the problem at hand. In general, to classify

something as an SoS, it must have the preponderance of the characteristics described. There are certainly examples at the extremes—a system is most clearly either a singular system or most clearly an SoS, but there are equally certainly systems that exist in the grey area in between. The point of classifying systems is to help identify what techniques and perspectives may or may not be useful for a given problem.

An example of the distinction between a system of sub-systems and an SoS clarifies the issue. A system of sub-systems is a jet fighter. It contains many sub-systems such as the weapons system, avionics, engine, and so forth, each of which are their own system; however, these systems are more properly seen as sub-systems since they do not perform a useful activity if isolated from the other sub-systems. In general, the collection of sub-systems does not generally exhibit the characteristics of an SoS. On the other hand, one could consider an aircraft carrier, complete with its full complement of aircraft and other supporting activities. While this, in one sense, is a singular unit that operates autonomously, and may be considered a singular system with many sub-systems, it can equally be considered an SoS, as each sub-system or constituent system can perform an independent, useful action (e.g., the aircraft, the ship). In this sense, an aircraft carrier may be both an SoS and a singular system of sub-systems. The choice of classification depends upon the purpose of the analysis.

c. SoS Classification

SoS are classified by the amount of central control and agreed upon purpose of the SoS. Maier (1998, 278) categorizes SoS as: “virtual,” “collaborative,” or “directed.” The DOD (2008, 4–5) classifies SoS similarly with the addition of an “acknowledged” category. Most DOD SoS programs are acknowledged SoS (DOD 2008). This dissertation only addresses acknowledged or directed SoS; the other categories are included for completeness.

(1) Virtual Systems of Systems

A virtual SoS lacks central control and an agreed upon purpose. An example of a virtual SoS would be a free-market economy (DOD 2008).

(2) Collaborative Systems of Systems

A collaborative SoS maintains a central purpose but lacks centralized control. An example would be the World Wide Web (DOD 2008).

(3) Acknowledged Systems of Systems

An acknowledged SoS has a central purpose and partial central control, in the sense that there is an entity charged with ensuring the SoS's success, but that entity may not have coercive or budgetary power over its constituent systems. An example would be the U.S.'s ballistic missile defense system (DOD 2008).

(4) Directed Systems of Systems

A directed SoS is both centrally controlled and has a centralized purpose. It remains an SoS because its constituent systems may still be able to make independent managerial choices, so long as they do not negatively impact the SoS and are operationally viable independent entities, though they have been designed to operate in the context of the SoS. Furthermore, a directed SoS meets the other three criteria of evolutionary development, emergent behavior, and geographic dispersion. An example is the ill-fated Army FCS (DOD 2008).

2. Systems Engineering versus Systems of Systems Engineering

The characteristics of SoS and implications for SoSE reach across all aspects of systems engineering, including management, design, integration, and operations. Giachetti (2014) concisely captures the essence of the distinction between the two domains in Figure 9.

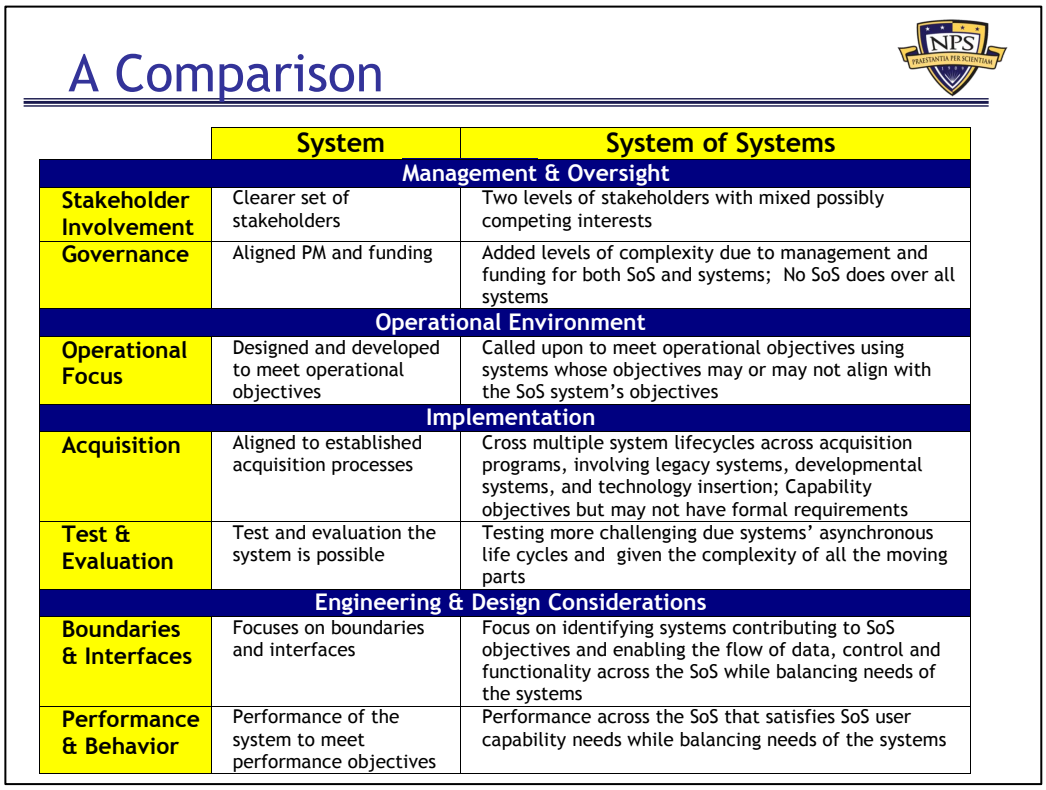


Figure 9. Comparison of Systems and SoS Engineering.
Source: Giachetti (2014)

In particular, the engineering and design of an SoS must balance the needs of constituent systems and the SoS as a whole in a "win-win" manner. This is particularly distinct from the traditional systems engineering top-down methodology in which top-level functions are identified and subsequent analysis follows a traceable train of logic from need to function to form. SoS, on the other hand, necessarily must start with existing systems and be developed both top-down (i.e., function to form) and bottom-up (i.e., form to function) to achieve the balance between SoS and constituent level system requirements. As a result of these differences, practitioners have developed SoSE models to capture these differences. These include the "trapeze model," the "wave model," the "iterated vee," and Sage and Biemer's SoS Engineering Process.

The "trapeze model" is called the "Core SoS SE Elements and Their Relationships" by (DOD 2008) and seen in Figure 10. It demonstrates the many interrelationships that must be understood to assess and engineer an SoS. The seven Core

Elements: “Translating Capability Objectives,” “Understanding Systems and Relationships,” “Assessing Performance to Capability Objectives,” “Developing and Evolving an SoS Architecture,” “Monitoring and Assessing Changes,” “Addressing Requirements and Solution Options,” and “Orchestrating Upgrades to SoS” describe the various necessary activities for SoSE per the DOD (2008). These provide a useful conceptual framework for SoSE, but are generally unwieldy as a repeatable process that produces predictable results.

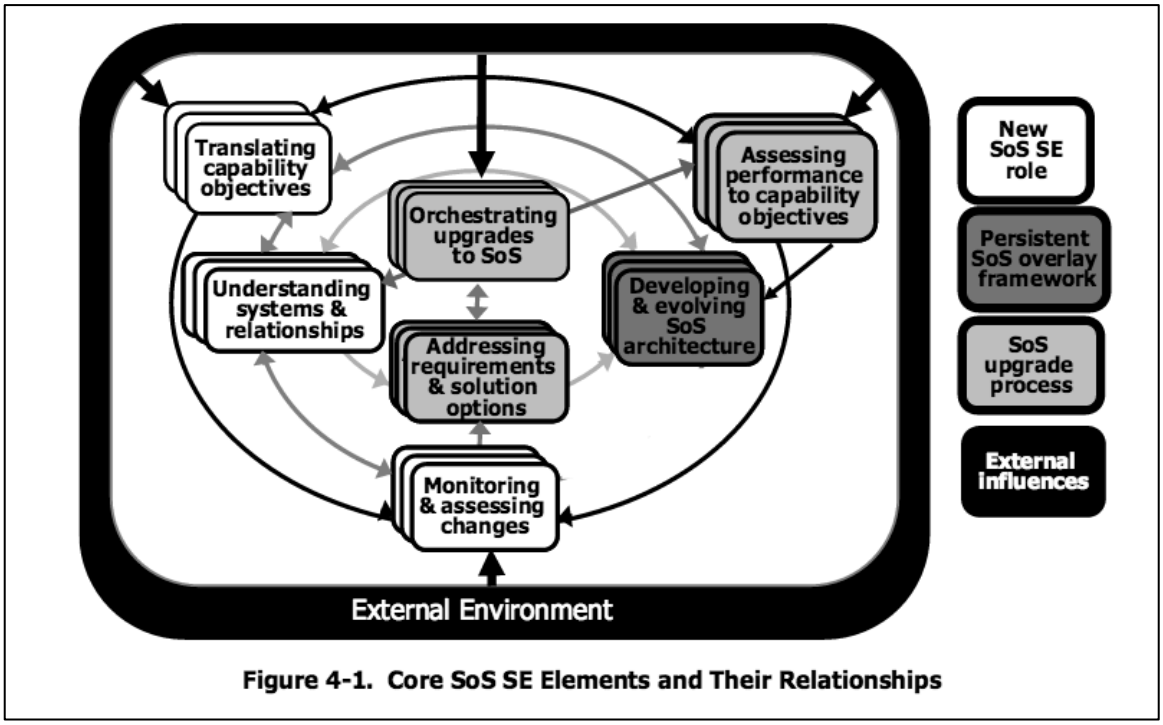


Figure 10. “Trapeze Model.” Source: Department of Defense (2008)

To address some of the limitations of the “Trapeze Model,” Dahmann et al. (2011) developed the “Wave Model” seen in Figure 11. This model takes the DOD’s seven “Core Elements” and places them in an iterative, repeatable model. This model combines the elements of “Translating Capability Objectives,” “Understanding Systems,” “Assessing Performance Against Objectives,” and “Monitoring Change” into a single concept, “Conduct / Continue SoS Analysis.” This effectively is the step in SoSE in which desired emergent properties are defined and assessed according to SoS

performance. This must be repeated continuously as strong or spooky emergence, or non-predicted simple or weak emergence may arise with changes to the SoS. The subsequent steps are fairly self-explanatory and map directly to their corresponding "Core Elements" as seen in Figure 11.

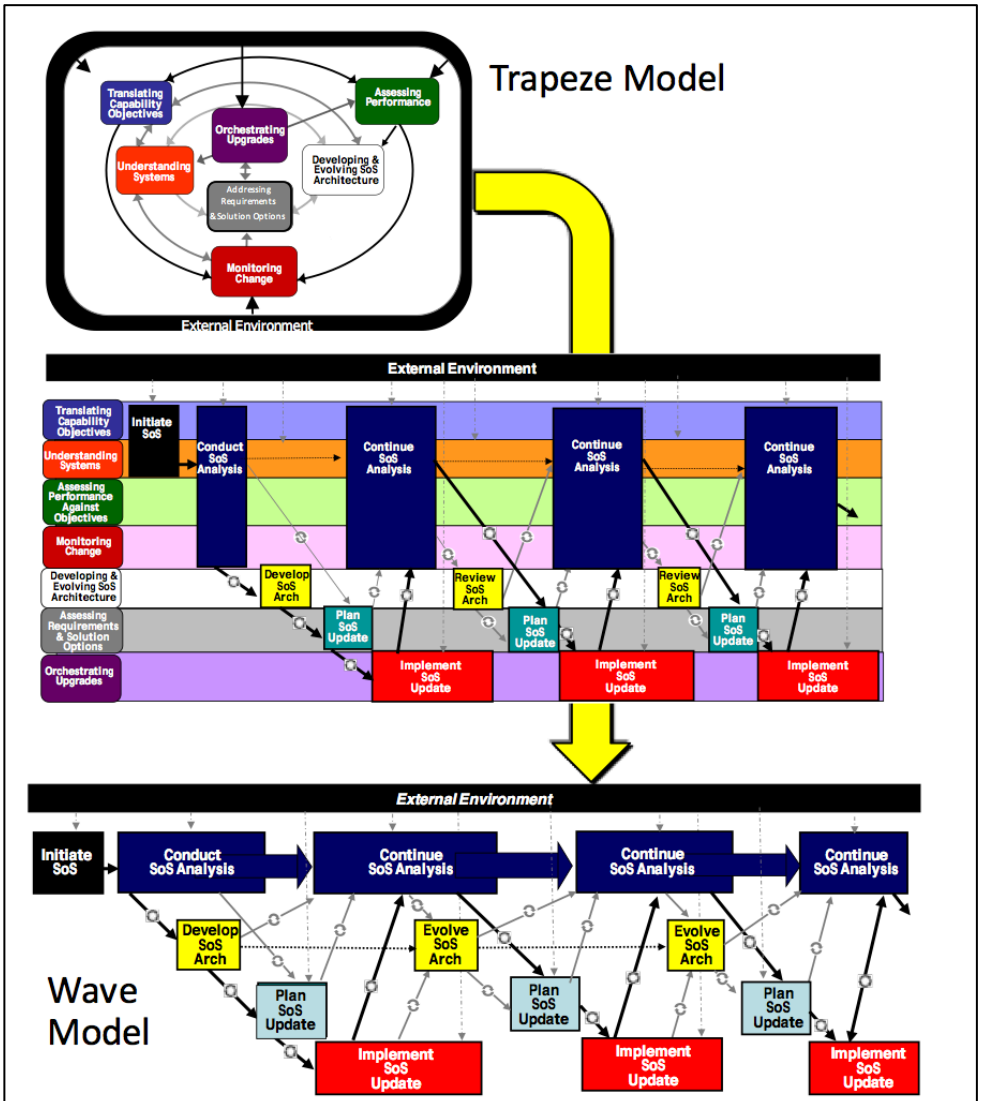


Figure 11. The Wave Model. Source: Dahmann et al. (2011)

SoSE practitioners have developed a somewhat more detailed "Iterated Vee" that is analogous to the typical systems engineering vee model. The DOD (2008) version of this iterated vee is seen in Figure 12. This model emphasizes the necessity to conduct upfront SoSE before conducting system level engineering. In this case, much of the engineering process is similar to typical systems engineering—identify the SoS problem and requirements, identify the necessary functions that must interact to provide useful emergent properties, identify potential physical systems that can meet these functions (i.e., constituent systems), and develop solutions that will cause these interactions to occur and be favorable to the constituent systems.

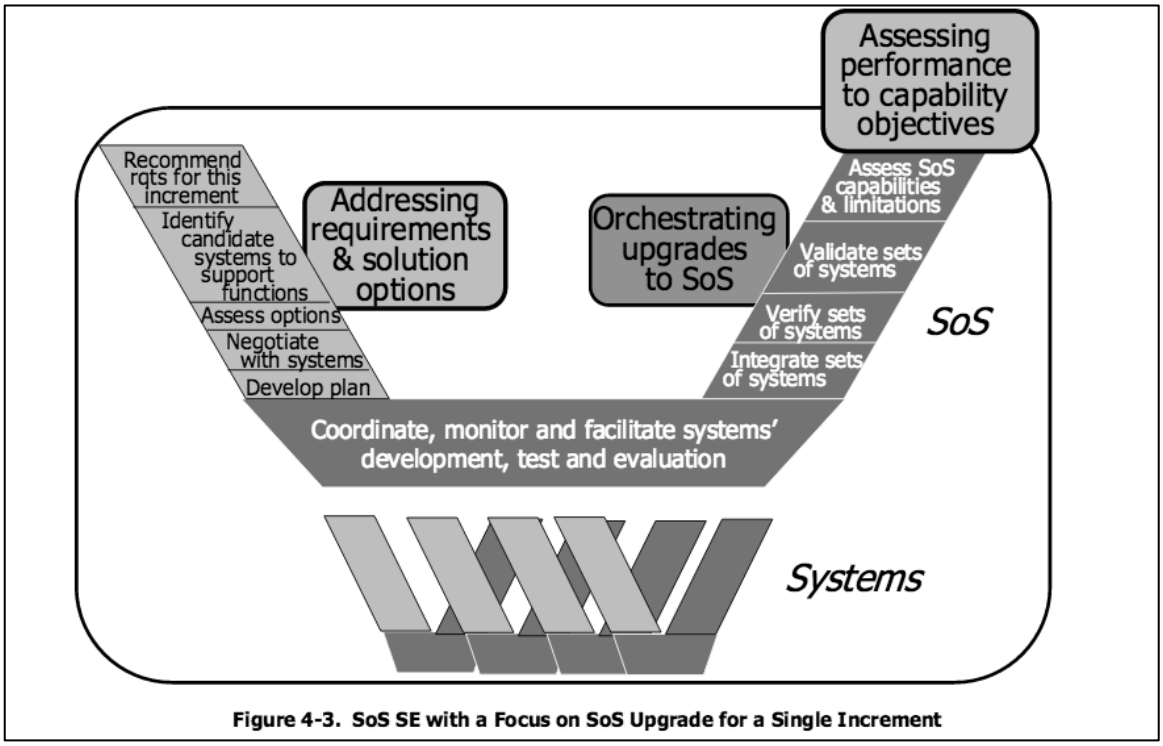


Figure 12. Iterated Vee Model. Source: Department of Defense (2008)

The final SoS engineering process was developed by Sage and Biemer (2007) and is seen in Figure 13. This figure is somewhat more complex than the preceding figures, but it encompasses much of the same information. Importantly, it identifies the various levels of SoSE identified as "Enterprise Activities," "Development Activities," "Operational Activities," and "Technical Activities" (Sage and Biemer 2007) and the

links among these different types of activities. This shows how SoSE operates at a key intersection of high level, strategic enterprise engineering, the technical aspect of system development and integration, along with management and operation of the systems and SoS. Sage and Biemer note that there is necessarily significant iteration and simultaneous activity in this process and that there are many more links among the various activities than displayed, but to display all of them would obscure the figure.

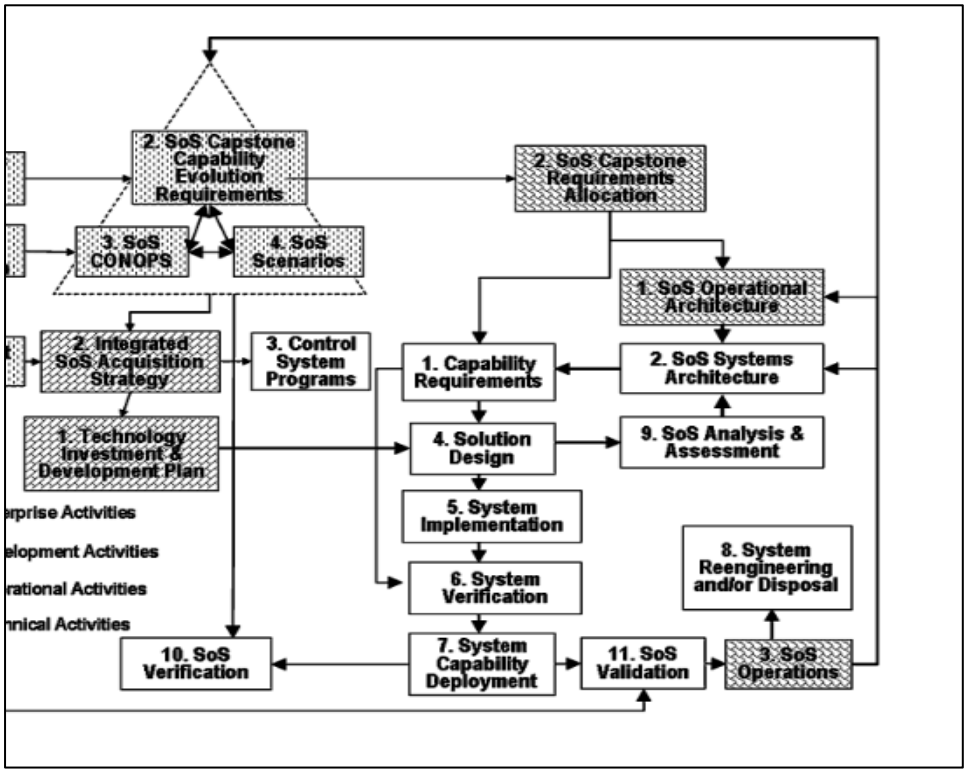


Figure 13. Sage and Biemer SoS Engineering Process.
Source: Sage and Biemer (2007)

In the preceding four SoSE models, it is clear that there is a continuous, iterative nature to SoSE. Within this, there occurs a periodic design phase in which SoS engineers design or modify interactions that can elicit desired emergent properties and, possibly, react to unpredicted emergent properties. This design phase consists of SoS analysis and SoS architecting. Figure 14 highlights where this design phase occurs within SoSE. SoS design, in a MBSE environment is the focus of this research and the topic of the following section.

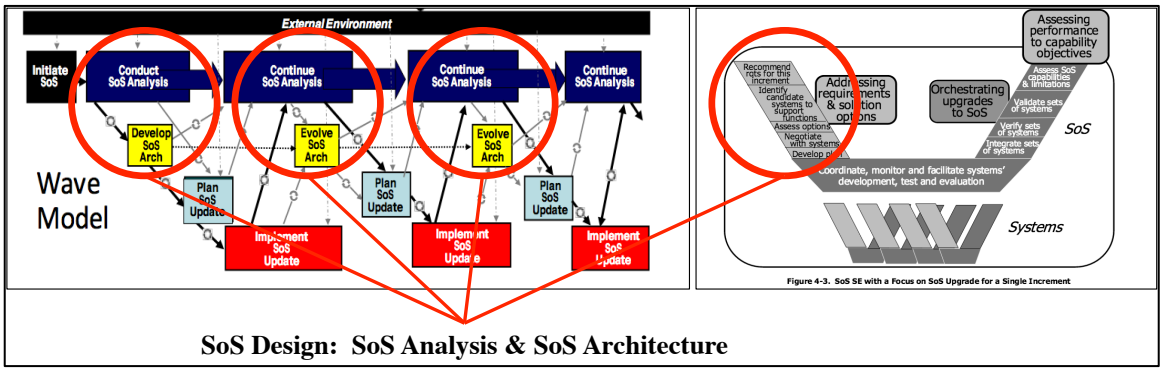


Figure 14. Where SoS Design Occurs in SoSE. Adapted from Dahmann et al. (2011) and Department of Defense (2008)

3. Conclusion

SoS are unique in that they are composed of operationally and managerially independent systems that interact to produce a desired emergent behavior. That the constituent systems are independent—they make decisions, respond to inputs according to their needs, and are not controlled by the SoS—has implications upon how they must be architected. Consideration must be accorded to not merely the technical, but relationships and methods by which these systems interact. This is expressed in the SoS architecture. Furthermore, the potential complexity of SoS operation mandates unique requirements for their analysis. Combined, these affect how one must design SoS.

E. SYSTEM OF SYSTEMS DESIGN

SoS design is the process by which an SoS architecture is realized. A SoS architecture must represent the unique SoS features—physical composition, processes, and organization. These features affect how SoS are analyzed. Together, these unique qualities make SoS design, particularly when framed in the context of tradespace exploration, a unique, and open, question.

1. System of Systems Architecture and Architecting

Maier and Rechtin define a (systems) architecture as, "The structure—in terms of components, connections, and constraints—of a product, process, or element" (Maier and Rechtin 2009, 423). They further elaborate this as: a matter of synthesis and analysis,

engineering and art, which ties human needs to system possibilities (Maier and Rechtin 2009). Architectures may be described in many ways; there are a variety of architecture frameworks that prescribe necessary elements of a system architecture. More generally, an architecture is only complete if it describes all of the various views necessary to understand a system (Maier and Rechtin 2009).

a. Systems Architecture and Architecting

Much has been written regarding systems architecture and architecting (Maier and Rechtin 2009; Buede 2000; Blanchard and Fabrycky 2011). For the purpose of this dissertation, we shall consider systems architecting in the common trichotomy of functional, physical, and allocated architectures.

A functional architecture describes what a system is supposed to do (Buede 2000). This is typically expressed as a functional hierarchy and augmented by functional flow block diagrams or IDEF0 diagrams (Buede 2000). More importantly, the functions of a system necessarily support the system objectives and are traceable to those objectives.

A physical architecture describes the components of a system that will complete the functions (Buede 2000). These may be systems (in the case of an SoS), sub-systems, components, or configuration items, depending upon the level of detail of the architecture. This may be represented as a hierarchy and be generic or instantiated representations (e.g., a plane versus an F-22) of physical components (Buede 2000).

The allocated architecture (formerly called operational architecture) ties the functional (what) to the physical (who) to describe how the system completes its objectives (the how) (Buede 2000). Importantly, one must allocate functions to physical components as seen in Figure 15. Buede (2000) argues that the most effective and preferable way to do this is through a bijection, where one function is linked to one component.

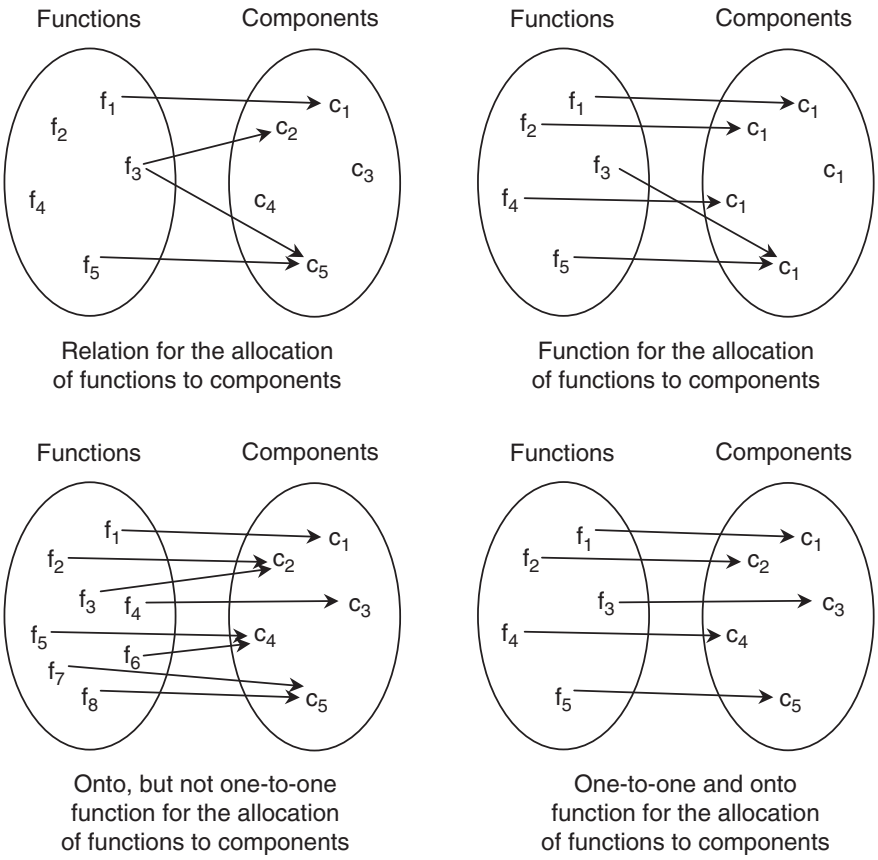


FIGURE 9.3 Mathematical relations and functions for the allocation of engineering functions to components.

Figure 15. Allocation of Functions to Components. Source: Buede (2000)

Finally, note that within the field of systems architecting, there are a number of architecture frameworks that describe and standardize how system architectures are to be developed, described, and their content (Maier and Rechtin 2011). Two of the most common architecture frameworks are the DOD Architecture Framework (DODAF) and Zachman Framework. These have been described in detail in by many researchers, e.g., (Dam 2006; DOD 2011; Giachetti 2010). This research references DODAF; however, this is as it is useful for the practical demonstration, any relevant framework may be used for a particular application. For a discussion of DODAF and its various views, see Appendix A.

Systems architecting is a key, if not *the* key, aspect of systems design. It ties human needs and desires to engineering reality. It is both prescriptive and descriptive in demonstrating what the system should and can do. The process of architecting is an inherently iterative one that cycles through creativity and analysis, desirability and feasibility. Architecting, particularly of complex systems, is enhanced by MBSE tools and methodologies along with architecture frameworks to facilitate communication and highlight consistency and traceability across an architecture.

b. System of Systems Architecture and Architecting

In some regard, architecting an SoS is no different than architecting a system. Fundamentally, the goal is the same, to link human needs with engineering potential, to describe the design of the system within the bounds of the "-ilities," and to define a manageable engineering problem. SoS architectures do have unique needs, however; in particular, they must consider the physical architecture as related to the constituent systems, the processes that regulate how systems may interact, and the organization that defines constituent system relationships.

(1) "Architecting Principles for Systems-of-Systems"

Maier's (1998) seminal article,[8] "Architecting Principles for Systems-of-Systems" details the definition and categories of SoS and outlines key heuristics for architecting them. His definition and categorization of SoS was referred to in Section II.D.1. Maier (1998) argues that SoS are architecture centric, specifically, information interface architecture centric. This is a direct result of the geographic dispersion and independence of the constituent systems. He states his analysis as follows:

> Since the components are often developed independently of the aggregate, the aggregate emerges as a system in its own right only through the interaction of the components. Because elements will be independently developed and operated, the system-of systems architect must express an overall structure largely (or even wholly) through the specification of communication standards. (Maier 1998, 268).

[8] As of this writing, Maier's article is cited by over 1,000 others on Google Scholar.

Combining this observation with the fact that SoS develop evolutionary according to the changes of constituent systems, Maier presents four SoS architecting principles: "Stable Intermediate Forms," "Policy Triage," "Leverage at the Interfaces," and "Ensuring Cooperation" (Maier 1998). These principles are meant to demonstrate best practices or heuristics for engineering an SoS.

"Stable Intermediate Forms" is a heuristic that recommends intermediate systems be capable of achieving useful purposes before the entire system is brought into being (Maier 1998). Applied to SoS, this means that the SoS may continue to exist if an individual system leaves, moreover, the loss of a single constituent will not be so catastrophic as to cause other constituents to leave the SoS (Maier 1998). This is necessary as constituent systems have operational and managerial independence, and, as such, may leave the SoS for any variety of reasons.

"Policy Triage" is a heuristic that invokes the concept of medical triage: only help those who can be helped and cannot recover without help, ignore the others (Maier 1998). For SoS, the implication is that one must attend to what one can control, namely the interfaces among the constituent systems, and not the internal workings of the systems themselves. Maier puts this aptly as, "The design guidance is to choose very carefully what to try and control. Attempting to over control will fail for lack of authority. Under control will eliminate the system nature of the integrated system" (Maier 1998, 273). In an SoS, an engineer must clearly identify what he can and what he cannot engineer.

"Leverage at the Interfaces" is a heuristic that directly applies the previous one. As Maier argues, an SoS engineer can only control the interfaces; he must focus his architecture at that level. In fact, Maier makes a somewhat bold claim:

> When the components of a system-of-systems are highly independent, operationally and managerially, the architecture of the system-of-systems *is* the interfaces. There is nothing else to architect. (Maier 1998, 273)

This claim is certainly true for collaborative or virtual SoS; it arguably has applicability to acknowledged and even directed SoS. Certainly no SoS architecture can be complete without a thorough description of the interfaces among the constituent systems; however, in the case of acknowledged and directed SoS, the SoS program has

some greater operational and managerial control which requires architecting, i.e., non-material aspects such as processes and organizations.

The final heuristic, "Ensuring Cooperation," speaks to the independence of the constituent systems. In all SoS, the constituent systems choose to participate or not, at least to a degree, depending upon the type of SoS (Maier 1998). As such, the motivation to participate must be factored into the design of the SoS (Maier 1998). There are a variety of means of doing this, and will vary with the distinct nature of the SoS, but this principle must be accounted for in architecting an SoS.

(2) Subsequent SoS Architecting Research

Maier's research along with a growing need for SoS engineering and architecture, prompted further research. Cole (2008) provides a comprehensive review of SoS architecture. He presents four SoS architecture design principles: "Needs Often Compete," "Needs Change Over Time," "Resource Availability Constrains the Solution Space," and "Design Compromise is Necessary" (Cole 2008, 45-47). In this context, the needs are those of the constituent systems and the SoS. The titles are self-explanatory; the point, similar to Maier's heuristics, is that one must focus on how the constituent systems interact physically. In Cole's work, these interactions are framed as needs. Cole further articulates SoS architecting with his six "SoS Architecture Considerations:" Autonomy, Complexity, Diversity, Integration Strategy, Data Architecture, and System Protection (Cole 2008, 47–55). Importantly, Cole outlines two strategies for system integration, bridging and refactoring. Bridging involves developing a new system that can interface with existing systems with only minor modification to existing systems. Refactoring is conducting potentially significant modifications to existing systems so that they can interface directly. These two strategies are seen in Figure 16.

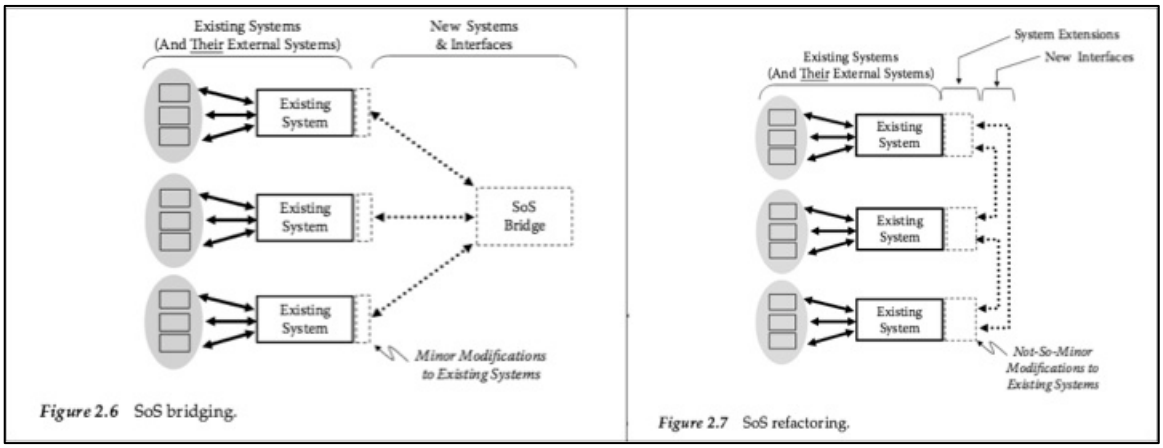


Figure 16. Cole's SoS Architecting Strategies. Source: Cole (2008)

Cole further describes three types of data architecture strategies for SoS. First, note that a data architecture is a representation how data is stored, transmitted, and understood across a system. This is not unique to SoS engineering; for example, it is used in enterprise engineering (Giachetti 2010). While data and information architecture is important in engineering many systems, it is particularly important to SoS as, per Maier's description, SoS information interface architecture *is* the SoS. Cole describes three data architecture strategies: uncoordinated, coordinated, and federated as seen in Figure 17. It is important to note that sharing information among different systems is particularly difficult as not only must one physically transmit the information, the information must be "usable" among the different systems. There must be semantic interoperability, such that System 1 may understand and use System 2's information. How an engineer architects this is highly important to developing an SoS.

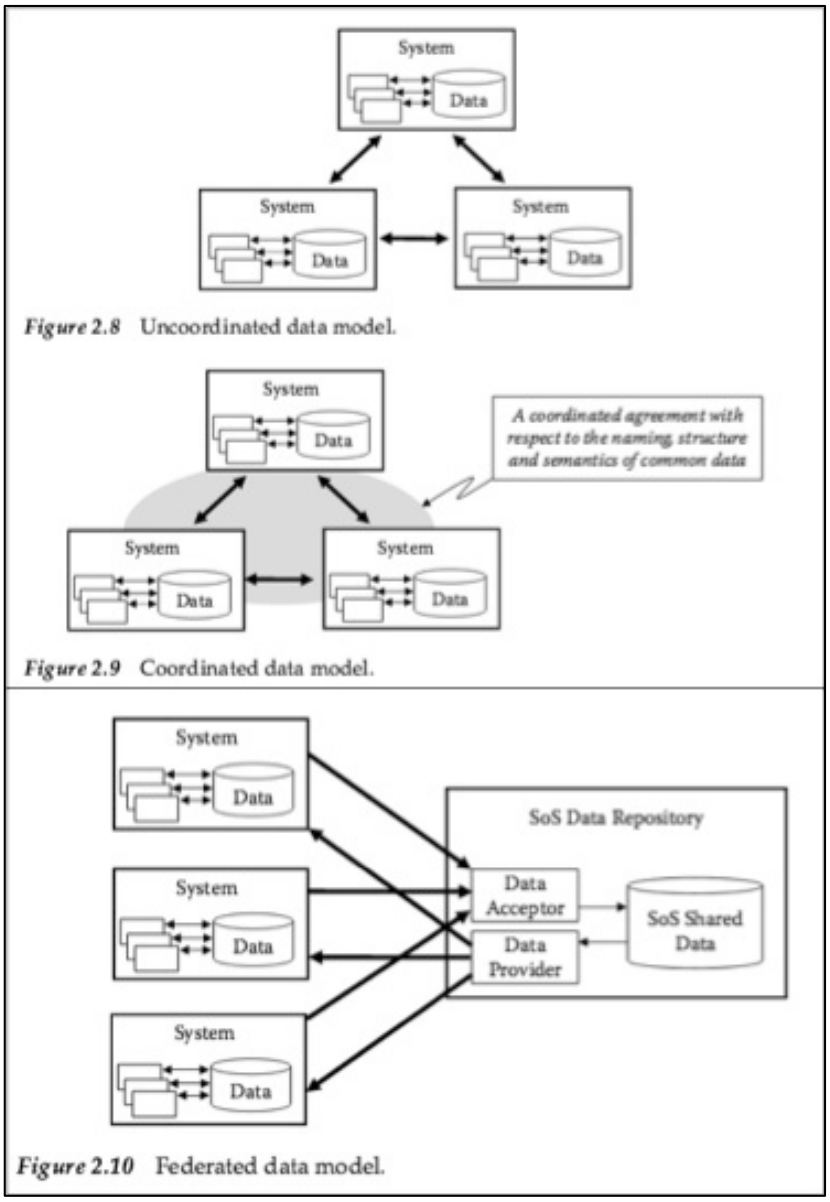


Figure 17. Cole's Data Architecture Models. Source: Cole (2008)

Dagli and Kilicay-Ergin (2009) outline their perspective on SoS architecting. Importantly, they compare system and SoS architecting, as seen in Table 1. One can note that this table outlines that much of the focus of SoS architecting is at the "meta-level" and is focused on how interactions among software, people, and systems occur and the interfaces that encourage these interactions.

Table 1. SoS Architecting versus Systems Architecting.
Source: Dagli and Kilicay-Ergin (2009)

	System of Systems Architecting	Systems Architecting
Architecting properties	▪ Abstract, meta-level ▪ Fuzzy uncertain requirements ▪ Network-centric ▪ Software intensive ▪ People intensive ▪ Intensive communication infrastructure ▪ Network of various stakeholders ▪ Collaborative emergent development ▪ Dynamic architecture	▪ Domain specific systems level ▪ Several stakeholders ▪ Controlled development ▪ Static architecture
Architecting constraints	▪ The same classical systems architecting processes, but at the meta-level ▪ Emphasis is on interface architecting to foster collaborative functions among independent systems ▪ Concentration is on choosing the right collection of systems to satisfy the requirements ▪ Scalability ▪ Interoperability ▪ Trustworthiness ▪ Hidden cascading failures ▪ Confusing life cycle context	▪ Architecting processes at component and systems level ▪ Monolithic systems architecting (optimize individual systems) ▪ Concentration is on building the right physical technical architecture ▪ Clear life cycle context
Legacy systems	▪ Abstraction level determines the integration of legacy systems to other systems ▪ Large amount of variety of legacy systems	▪ Integration of legacy system to system components are more clear compared to SoS
Architecting tools	▪ Model-centric and executable models ▪ Balance of heuristics, analytical techniques and integrated modeling	▪ Document-centric frameworks ▪ Model-Centric frameworks ▪ Pure analytical techniques ▪ Heuristics

Maier and Cole both devote significant effort to detailing the necessity of an SoS architecture to satisfactorily integrate different constituent systems via an information architecture. This is, of course, highly important. In some sense, this is the physical architecture of the SoS. Similarly, Dagli and Kilicay-Ergin focus on designing interfaces to encourage specific physical systems to interact. However, this focus is somewhat exclusive of functional and allocated SoS architecting.

(3) Distinctions Between Systems Architecting and SoS Architecting

SoS architectures must describe both the composition of the SoS, the constituent system interfaces, and the means by which their interactions are governed to produce the desired emergent behaviors. This requires both systems (technical) and enterprise (non-technical) perspectives. This is because SoS are composed of independent constituent systems that make decisions regarding SoS participation and their operational activity.

The physical architecture of an SoS is the composition of the included constituent systems and the technical description of their interfaces. These are described in DODAF by both the SV-3 and DIV-1, DIV-2, and DIV-3 views (DOD CIO 2010). Much of SoS engineering is devoted to choosing the composition of systems (Chattopadhyay 2009; Mokhtarpour and Stracener 2014) and the technical interface architecting (Maier 1998; Cole 2008; Biltgen, Ender, and Mavris 2006). This is warranted, as it is both a difficult problem and a necessary first step in the architecting process. A collection of systems with an inability to interface cannot be an SoS.

A SoS has a functional architecture; it describes what the SoS does. These functions, at the highest level, are the result of a desired emergent property of the SoS. That is, if one desires an SoS to perform a given function, one must induce systems to interact in a manner so as to provide that functionality. If a single system can provide that functionality, the problem is complete and a matter of systems engineering (this is not trivial, but outside the scope of this research). This is modeled in Figure 18. Note that a single system may have multiple types of interactions with different systems.

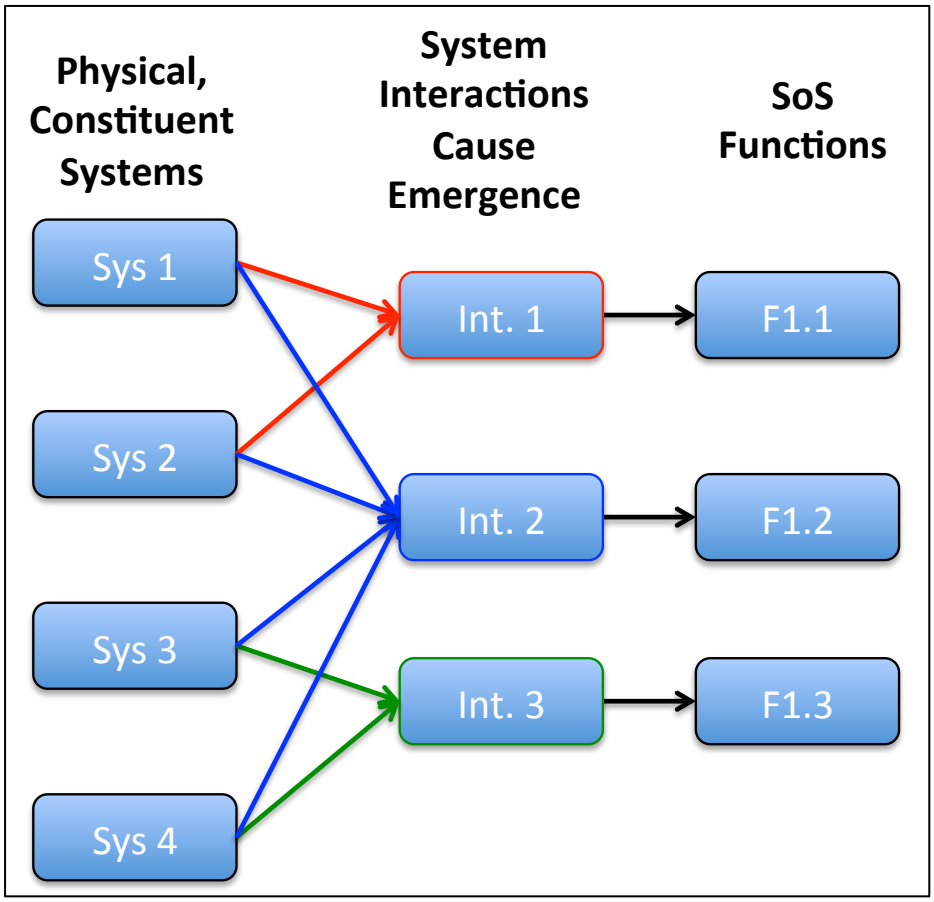


Figure 18. SoS Interactions Provide SoS Functionality

These interactions are not simply a matter of physical interfacing, rather, they are a function of multiple systems sequencing their activity and modifying their activity according to the actions of the others. This requires an enterprise perspective—organization and process views (Giachetti 2010).

Each constituent system provides some level of functionality, capability, or operational activities. Within DODAF, a system's capabilities are described by the various Capability Viewpoints (DOD CIO 2010); see Appendix A for further details. Moreover, the SV-4, SV-5a and SV-5b provide greater detailed descriptions of these capabilities (DOD CIO 2010). Using DODAF standardizes the language of capabilities and functions so that one may establish parity among the different system-level architecture descriptions. More to the point, any emergent behavior is a product of these functionalities. Logically, an SoS may only achieve a desired emergent behavior if its

constituent systems contain all of the necessary functionality. For each desired emergent behavior, one must be able to describe, either through a functional flow or a set of rules (in DODAF, the OV-5 and OV-6 models), how an emergent behavior occurs. In the cases of simple emergence (readily understood and modeled), this is most easily described by a functional flow; in the cases of weak emergence (understood and possible to be modeled after observing it), this is more likely to be modeled using rules governing interactions. Regardless, an SoS requires a description of the processes that govern the interactions.

Constituent systems are independent, decision-making entities. Moreover, with the exception of fully autonomous systems, people operate the constituent systems. Accordingly, an SoS is not simply a technical system, but also an organization. There is a diverse range of literature regarding the study of organizations (e.g., March and Simon 1958; Galbraith 1977; Daft 1998; Burton, DeSanctis, and Obel 2006). Organizations are defined similarly to systems, except that they are social entities as opposed to technical ones; this is articulated as, "organizations are made up of people and their *relationships* with one another. An organization exists when people interact with one another to perform essential functions that help attain goals" [Emphasis added] (Daft 1998, 11). Importantly, it is these relationships that must be well defined in an organization to influence the behavior of the constituent members (March and Simon 1958).

Typically, organizational design, particularly with regard to a business, is concerned with the totality of an organization—its goals, measures of performance, processes, people, and coordination (Daft 1998; Burton, DeSanctis, Obel 2006). This significantly overlaps with much of systems engineering; accordingly, for this dissertation, organizational design only refers to the structure and definition of the relationships among the constituent systems of the SoS.

Traditionally, organizational structures are defined according to the information and decision-making affects relationships have between the various entities of the enterprise (Burton, DeSanctis, Obel 2006). There are a variety of organization structure types: simple hierarchy, functional, divisional, matrix that vary groupings of people within the enterprise according to their rank, their function, their market (type or location), or some combination thereof (Burton, DeSanctis, Obel 2006; Giachetti 2010).

The structure may be expressed as a set of relationships (or a matrix) between the entities in the organization and the corresponding definition of those relationships; this view of an organization generally coincides with the OV-4: Organizational Relationships view in DODAF (DOD CIO 2010). A well-defined organizational relationship articulates requirements for communication and decision-making; e.g., in the Army, there are Commander's Critical Information Requirements (CCIR) that detail information a subordinate must pass to the commander (U.S. Army 2006).

A SoS is both technical and non-technical; accordingly, its architecture must represent both of these perspectives. At a minimum, an SoS architecture should include a physical description of the constituent system composition and their interfaces, the process(es) by which the SoS achieves its emergent behavior, and the organization that defines the relationships among the systems. Together, these both describe and prescribe SoS activity in a complete manner that may be both used for SoS assessment (in a model or simulation) and SoS realization.

2. System of Systems Analysis

The analysis of an SoS is similar to the analysis of any system and differs primarily in the details of how it is done. Gibson, Scherer, and Gibson (2007, 29) list six major phases of systems analysis: "1. Determine goals of the system." "2. Establish criteria for ranking alternative candidates." "3. Develop alternative solutions." "4. Rank alternative candidates." "5. Iterate." and "6. Action." This is generally in line with other texts on systems analysis such as (Blanchard and Fabrycky 2011; Buede, 2000). These steps may be somewhat simplified as problem definition (including steps one and two), analyze systems (including steps three and four), and implementation (steps five and six).

a. System of Systems Analysis Problem Definition

Defining a systems analysis problem involves determining the goals of the system and the means by which to compare alternative solutions. This is typically expressed in terms of functions and functionality and through MOEs and MOPs. Note that an MOE is a measure of how successful the system operation is relative to the need and an MOP is a measure of how well a system operates according to its design (Parnell et al. 2011). Much

work has been written regarding problem definition and MOE and MOP selection, e.g., (Parnell et al. 2011; Blanchard and Fabrycky 2011).

The goals of an SoS are necessarily realized through emergent properties. Accordingly, SoS analysis problem definition should be focused on how the SoS performs these emergent functions. The MOEs and MOPs selected should support these fundamental SoS objectives in a clear and logical manner. Importantly, they should be focused on the aspects of the SoS that the engineer has control over. Examples of MOEs might include various measures of overall (SoS) mission accomplishment such as time to mission accomplishment, force exchange ratio, or similar total SoS measures. Examples of MOPs might include measures of connectivity of a designed interface, percent of available systems willing to participate using a given interface, or reliability of an interface. It is inappropriate for an SoS MOE or MOP to be focused on a constituent system level property or function; system-level engineers more appropriately answer such questions. Finally, the thresholds (minima or maxima) and goals of a given performance measure and their associated values vary according to decision-maker preferences. In the context of exploratory analysis, the question of defining these specifically *a priori* is less important than defining the relevant measures as it is assumed that these thresholds and goals may change during TSE.

b. How to Analyze a System of Systems

For a non-extant SoS, as is the case in SoS design, the typical method of analysis is to model the SoS and assess its performance of its various MOEs and MOPs in that model. This is no different than modeling for system assessment. Importantly in an SoS, one must capture the relevant perspectives of its design—its physical, process, and organizational view—as inputs and output its emergent behavior or other desired attributes.

The choice of model depends upon the system being modeled and the purpose for modeling that system. In the case of SoS, the typical purpose for modeling is to analyze and understand an emergent property. In this case, agent based models (ABM) are the most common choice, though Petri Nets, and Markov Chains, and Network Models have

also been used. Of note, Rainey and Tolk (2015) provide a comprehensive overview of modeling and simulation for SoS; Baldwin et al. (2015) provide an analysis of event based versus agent based simulation approaches for SoS.

Macal and North (2005) describe ABM as a model composed of agents with defined behaviors that interact with other agents and their environment; this gives rise to emergent behavior. This clearly is a useful way to approximate an SoS. Rainey and Tolk (2015) and Mour et al. (2013) provide multiple examples of using ABM for SoS. Giachetti et al. (2013) is another example of using ABM to assess SoS performance.

Petri Nets and Markov Chains are other common methods for modeling SoS. In both cases, there is a process flow, possibly stochastic, that mimics how SoS perform a fundamental objective. These are useful in cases where the interactions among systems are generally well understood, such as in the case of simple emergence. Wang (2007), Rao et al. (2008), and Kenley et al. (2014) provide examples of SoS analysis using Petri Nets; Giachetti (2015) is an example of using a Markov Chain for the same purpose. The advantage of such techniques is that they are less computationally intensive than ABM.

Networks that represent constituent systems as nodes and interactions as edges in a network are also useful for modeling an SoS. Garrett et al. (2011) use a network model to represent the Ballistic Missile Defense System [of Systems]. This work is useful in demonstrating how a network may represent an SoS, though it is flawed in that the subsequent analysis makes limited utility of their model. DeLaurentis et al. (2008) use traditional network measures (see, e.g., Newman 2010) to assess and enhance the Air Traffic Organization air route forecast. In general, network models using various network flow algorithms, such as presented by Ahuja et al. (1993), can be used to assess the performance of many metrics of an SoS represented as a network.

Perhaps more important than the specific choice of type of model, is that SoS cannot be well assessed through an aggregation of system level analysis. This, as with most aspects of SoS, stems from the fact that SoS present emergent properties and the interactions eliciting these properties must be included in the model. Anderson et al. (2013) demonstrate this with regard to SoS operational availability using the Sandia

National Laboratory SoS Analysis Tool (SOSAT). In this case, averaging the operational availability of the constituent systems is not a useful aggregation, as an SoS may be operationally available 100% of the time even if some of its constituent systems are not (due to the redundancy contained in the SoS).

Chattopadhyay (2009) present a method for combining attributes of systems for SoS. This is at odds with the preceding paragraph. Her method has three levels of "attribute combination complexity," low, medium, and high. Low-level combination is taking a best in class MOE or MOP for each constituent system and assigning it as the attribute of the SoS (Chattopadhyay 2009). Medium-level involves weighted averaging of system level attributes (Chattopadhyay 2009). High-level attribute combination is done through "data fusion" (Chattopadhyay 2009). While it is possible that the low and medium level attribute combinations can be useful in select cases, they generally fail for the reasons described in the preceding paragraph and are not generally useful for assessing emergent behavior. High-level combination through data fusion is useful, and though not done in the same way as ABM, it is a method of predicting emergent behavior through more complex combinations of system level attributes that mimic the system interactions. Despite these challenges, there may be instances where low or medium level attribute combination is useful, if a rough, first order level of analysis.

The actual analysis of an SoS is best-conducted using models that clearly represent the interactions among the constituent systems of an SoS and demonstrate emergent SoS behavior. These types of models include ABM, Petri Nets, Markov Chains, and Network Models. Lower level aggregation of constituent system level properties while computationally inexpensive, run the risk of presenting inaccurate SoS level properties and should be used with caution. The results of these models can inform decision-makers on the performance of SoS with regard to MOEs and MOPs and facilitate the choice of SoS design.

c. Challenges of SoS Modeling and Simulation

A side, but important, topic in SoS analysis is some of the outstanding challenges of SoS modeling and simulation. These challenges include model validation, model

integration, and the development of metal-models of SoS performance. These challenges impact SoS analysis and, accordingly, SoS design; in particular the development of meta-models.

(1) SoS Model Validation

SoS model validation is a challenge because it is often difficult, if not impossible, to conduct sufficient numbers of SoS experiments to assess the validity of a model. Particularly as SoS have the potential to constantly evolve, thus changing the assumptions of any model. Operational test and evaluation of an SoS is a challenge as it is often difficult to coordinate the activity of the operationally and managerially independent systems in a non-operational environment (i.e., a test scenario). Moreover, these tests are often difficult to reproduce to build sufficient data for a statistical analysis by which to validate the model. Accordingly, SoS models are rarely validated at the level of statistical analysis of repeated tests, rather they are validated with toy problems, face validity, or similar, lower level methods of model validation.

(2) Model Integration

Most systems within an SoS, being managerially independent, have pre-built, possibly validated models, of their performance. In the interest of economy and accuracy, it makes sense for an SoS model to incorporate these system level models. The challenge is that every model is built for a specific purpose and makes specific assumptions. These purposes and assumptions may not align well for the purpose of the SoS and across the various system models. Wang, Tolk, and Wang (2009) present the Levels of Conceptual Interoperability Model (LCIM) that outlines this problem with model interoperability rated across seven levels as seen in Table 2. Despite this problem, it is not impossible to overcome; Kewley and Wood (2012) present a case of a federated combat model to assess SoS performance of different combat systems demonstrating both the difficulty and possibility of federating different models to develop an SoS one.

Table 2. Levels of Conceptual Interoperability Model (LCIM). Adapted from Wang, Tolk, and Wang (2009).

Level	Layer Name	Information Defined	Capability
6	Conceptual	Assumptions, constrains etc.	High
5	Dynamic	Effect of data	
4	Pragmatic	Use of data	Medium
3	Semantic	Meaning of data	
2	Syntactic	Structured data	
1	Technical	Bits and bytes	
0	No	NA	Low

(3) SoS Meta-Models

The final major challenge in SoS modeling and simulation is in developing meta-models of the SoS. A meta-model, or response surface, is a model that is developed using various statistical techniques to return a response of interest from multiple variables (Montgomery 2005). It is developed through selective samples that are best chosen through a DOE. Montgomery (2005) provides an overview of basic experimental design; Kleijnen et al. (2005) provide a more detailed overview on advanced DOE techniques.

In this dissertation's notation, a meta-model is an approximation of a system attribute function, $f_a: \boldsymbol{D} \rightarrow \boldsymbol{R_a}$. A meta-model is an efficient way to define f_a as direct analysis of large numbers of design points is computationally intensive, if not impossible. Meta-models provide a reasonable approximation in much less time.

The challenge of meta-modeling and experimental design for SoS is that the experiments for SoS are highly complex, with many degrees of freedom, and, often, highly non-linear or even non-polynomial response surfaces (Kernstine 2013). Traditional methods of meta-modeling and DOE are currently inadequate for handling such response surfaces with significant higher order interactions between the variables (design parameters) (Sanchez and Wan 2012). In the case of an SoS, however, we

explicitly assume there are many higher order interactions among the parameters. Kernstine (2012) provides a solution to explore such spaces using adaptive sequential experiments. This is done through an algorithm that identifies significant areas of variance and explores them in greater depth (Kernstine 2012).

Kernstine's (2012) method is still insufficient for an SoS that is fully described by physical, process, and organizational parameters. For example, the network configurations formed by including or not including two or more of n potential systems may be considered one categorical variable with approximately 2^n levels. Furthermore, the number of different organizations and processes are also categorical in nature. So, while it is possible to define an experimental design for such variables (e.g. Vieira, et al. 2011), in this situation, the number of levels each parameter can take makes such designs unwieldy. Sanchez and Wan (2012) note that experimental designs to account for categorical variables are best when the number of levels each variable can take is 10 or fewer. In the case of an SoS defined across physical, organizational, and process parameters, this threshold is quickly surpassed. As an alternative, we develop a method to selectively choose a small sample of design points for analysis and only define an attribute function on that domain.

d. Conclusion

SoS analysis assesses an SoS design point for its system attributes. These attributes are, typically, the emergent behaviors of the SoS. To do this, one uses a variety of models and simulations, commonly ABM, but also Petri Nets, Markov Chains, and Network Models. The common means of approximating an attribute function, through DOE and meta-modeling, is problematic in the case of an SoS that introduces qualitative parameters (variables) with many levels (significantly greater than ten) and higher order interactions that are significant among them. This condition exceeds the threshold for contemporary MBSE methods, thereby creating a limitation in the state-of-the-art for SoS analysis.

3. System of Systems Design

SoS design is the process by which an SoS architecture is realized. This is done through identifying a set of possibilities and choosing among them. Methods of design decision-making include heuristics, normative, and exploratory. Researchers have provided SoS heuristics, normative methods, and limited exploratory methods as outlined in Figure 19. The challenges of SoS design—system complexity and competing perspectives challenge heuristics and normative methods and make SoS exploratory decision-making methods a useful alternative.

		Design Decision Making Methodology		
		Heuristics Decision Making	**Normative Decision Making** **Traditional SE**	**Exploratory Decision Making** **Design Theory / Tradespace Exploration**
Classification of Systems	**Monolithic Systems**	• *The Art of Systems Architecting,* Maier and Rechtin (2009)	• *The Engineering Design of Systems,* Buede (2009) • *Decision Making in Systems Engineering and Management,* Parnell and Driscoll (2011) • *Systems Engineering Analysis,* Blanchard and Fabrycky (2011) • *Defense Acquisition Guidebook Chapter 4, Systems Engineering*, DOD (2013) • *Systems Engineering Handbook,* NASA (2014) • *Systems Engineering Handbook*, INCOSE (2015)	• "Design Space Visualization and Its Application to a Design by Shopping Paradigm," Stump et al. (2004) • "The Tradespace Exploration Paradigm," Ross and Hastings (2005) • "Systems Engineering Resiliency: Guiding Tradespace Exploration within an Engineered Resilient Systems Context," Sitterle et al. (2015) • "Illuminating Tradespace Decisions Using Efficient Experimental Space-Filling Designs for the Engineered Resilient System Architecture," MacCalman et al. (2015) • "A Model-Based Systems Engineering Methodology for Employing Architecture in System Analysis," Beery (2016)
	Systems of Systems	• "Architecting Principles for Systems-of-Systems," Maier (1998) • "SoS Architecture," Cole (2008) • "System of Systems Architecting," Dagli and Kilicay-Ergin (2009)	• "A Robust Portfolio Optimization Approach to SoS Architectures," Davendraglingam and DeLaurentis (2015) • "A Conceptual Methodology for Selecting the Preferred SoS," Mokhtarpour and Stracener (2014) • "Synthesizing and Specifying Architectures for SoS," Kenley et al. (2014) • "Modeling and Simulation of Net Centric System of Systems Using Systems Modeling Language and Colored Petri-Nets," Rao et al. (2008) • "Systems Engineering Guide for Systems of Systems," DOD (2008)	• **"The System of Systems Tradespace Definition Methodology Through the System of Systems Architecture Feasibility Assessment Model," Gillespie (2016)** • "A Method for Tradespace Exploration of SoS," Chattopadhyay (2009) • "Development of a Collaborative Capability Based Tradeoff Environment for Complex System Architectures," Biltgen et al. (2006)

Figure 19. System Design Decision-Making Methodologies

a. SoS Heuristic Design

SoS heuristic design considerations (Maier 1998; Cole 2008; Dagli and Kilicay-Ergin 2009) were discussed in Section II.E.1. While useful, they require either normative

or exploratory augmentation, particularly when decision-makers are considering distinguishing between degrees of variation in system architectures.

b. SoS Normative Design

SoS normative design methods (Davendralingam and DeLaurentis 2015; Mokhtarpour and Stracener 2014; Kenley et al. 2014; Rao et al. 2008) have been the major thrust of SoS design research. These are useful in well-defined problems with clearly defined system attribute goals and thresholds. In general, however, these are all limited in that they only consider select aspects of an SoS architecture.

(1) Davendralingam and DeLaurentis, 2015

Davendralingam and DeLaurentis (2015) propose and demonstrate a method for analyzing SoS architectures by employing tools from operations research and financial engineering. They formulate the problem by imagining possible constituent systems as nodes in a network. Each system has input requirements and output capabilities; possible connectivity is established through connections in the network. Furthermore, a generalized method of SoS accomplishment is established as a (directed) network of capabilities. For example, the Ballistic Missile Defense System is represented as a network that links the capabilities of detect, track, intercept (Davendralingam and DeLaurentis 2015). With the problem established as such, the researchers applied methods of operations research and financial engineering such as mathematical programming to quantify the effects of adding a given system to the SoS so as to provide a set of Pareto optimal solutions balancing the risk associated with each system and capability added by each option. The authors applied this to a Naval Warfare scenario using various ships, communications packages, weapons packages, and aircraft to complete various missions. The subsequent analysis yielded a usable performance versus development time (risk) tradespace and data to facilitate engineering decision-making. The authors conclude their research with a call to examine nonlinear interactions and multi-decision-maker considerations for objective functions.

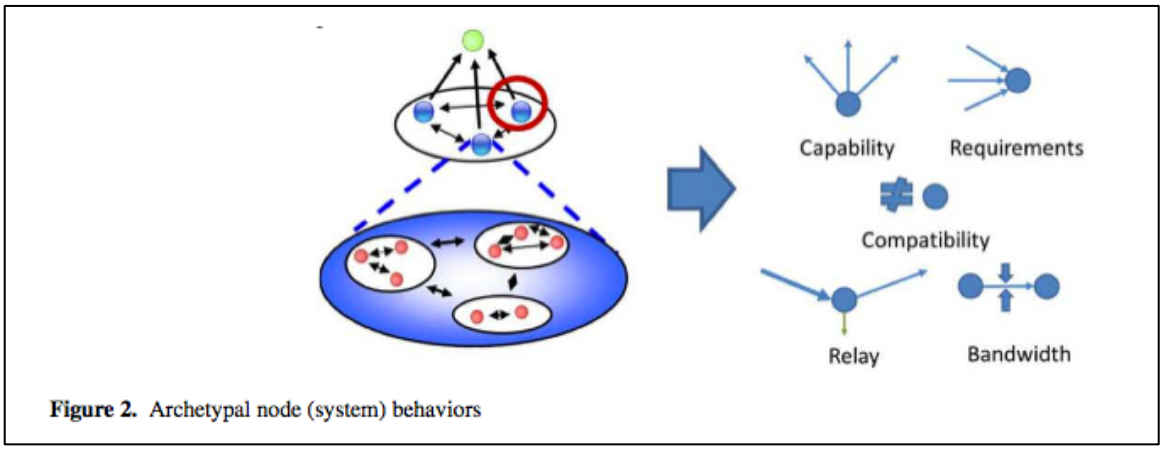


Figure 2. Archetypal node (system) behaviors

Figure 20. Davendralingam and DeLaurentis Archetypal SoS for Portfolio Optimization. Source: Davendralingam and DeLaurentis (2015)

This research is novel in that it presents a combinatorial approach to SoS development with regard to process architecting. The development and demonstration of analytic techniques for assessing the many possible combinations that can occur when developing an SoS from many constituent systems with overlapping capabilities is a useful aid to SoS designers. It is limited in that it only allows for a singular process architecture to achieve the desired emergent property. It is also limited as the objective functions to be optimized are set *a priori* and do not allow for trades in requirements to be made. This constrains the possible design space an engineer can consider as he architects the SoS. Finally, it does not explicitly identify that it is conducting an allocated architecture and consider how the different combinations of systems within the SoS may be affected by organizational allocations. Nor does it consider how the allocated architecture affects systems participation risk. Overall, this is a useful analytic technique that could be combined into a greater methodology and applied to specific problems.

(2) Mokhtarpour and Stracener, 2014

Mokhtarpour and Stracener, (2014) present a conceptual methodology for selecting systems to form an SoS. They included several key factors for assessing a general SoS: "Time to achieve SoS capability," "SoS mission reliability," "SoS basic reliability," "SoS operational availability," "SoS priority," and "SoS capability cost" (Mokhtarpour and Stracener 2014, 2). They subsequently formulated a general

methodology, seen in Figure 21. Each step is expanded upon, with an algorithmic process for steps one, two, and three; a combinatorial assessment for step 4; assessing the possibilities according to the metrics initially listed for step 5; and making a decision according to situation specific (i.e., the formulation of values and number of decision-makers) criteria for step 6. This methodology is quite systematic and serves as a useful guide for SoS decision-makers and planners.

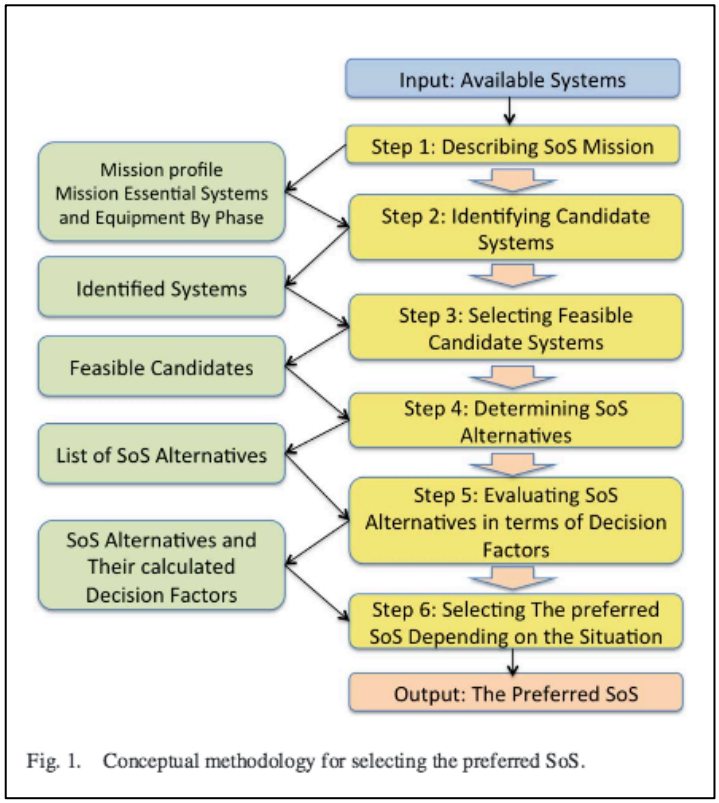


Figure 21. Conceptual Methodology for Selecting the Preferred SoS. Source: Mokhtarpour and Stracener (2014)

This methodology is one of the few such methodologies in the literature that take an analytic perspective on designing an SoS. It is clear, repeatable, and, though it not explicitly defined, it could conceivably be iterated. It is limited with regard to specific architecting; the article references what could be considered an SoS functional architecture through the use of a mission essential function list and mission essential systems in steps 1 and 2 (Mokhtarpour and Stracener 2014), though it does not

specifically identify this as a functional architecture for the SoS. The physical architecture is clearly the candidate systems chosen, though the methodology does not clearly allow for the development of new or modified interfaces, which greatly affect the feasibility of what systems are possible. The allocated architecture is not specifically mentioned, although the combinatorial aspect of Step 4 is a potential start of allocated architecting. It is limited in that it only considers a process view and does not consider organizational rules or policies that are the important architecture models that an SoS designer controls to realize these processes. The analysis methodology is useful in that it is fully described, but it inexplicably does not include operational performance as a measure (e.g., how well does the SoS complete the mission by any MOE); it includes the various "-ilities" and cost as the only drivers for assessment. These may or may not be the preferred measures for any given decision-maker. Overall, this methodology is useful in demonstrating the limitation of SoS design methodologies and a possible methodology for SoS design in very specific cases, namely directed SoS exhibiting simple emergence. It does not yield clear architecture models, it does not allow for tradespace exploration, nor is it integrated with contemporary MBSE methods.

(3) Kenley, Dannenhoffer, Wood, and DeLaurentis, 2014

Kenley et al. (2014) present a method that links common system architecting, with SoS specific characteristics, and MBSE techniques to specify SoS architectures. The process model they use is depicted in Figure 22. In particular, these are the first authors to explicitly state that the allocated architecture of an SoS is unique, stating, "Multiple possible allocated architectures can be defined from a functional and physical architecture. It is the primary goal of system of systems architecting to define feasible SoS architectures; to evaluate the ability of the architectures to satisfy mission requirements and the resources required to procure and operate the SoS" (Kenley et al. 2014, 3). The authors model the allocated architecture through a dynamics model, with functionalities acting as agents; in particular, they use the discrete agent framework developed by Mour et al. (2013). This automates the creation of possible allocated architectures allowing researchers to explore large design spaces. They further explore these architectures regarding their performance as judged through process flows modeled

in Petri Nets. These are dynamically linked to UML (and, by extension SysML) products. This automated synthesizing of network architectures combined with process flows allows a more full exploration of possible SoS designs.

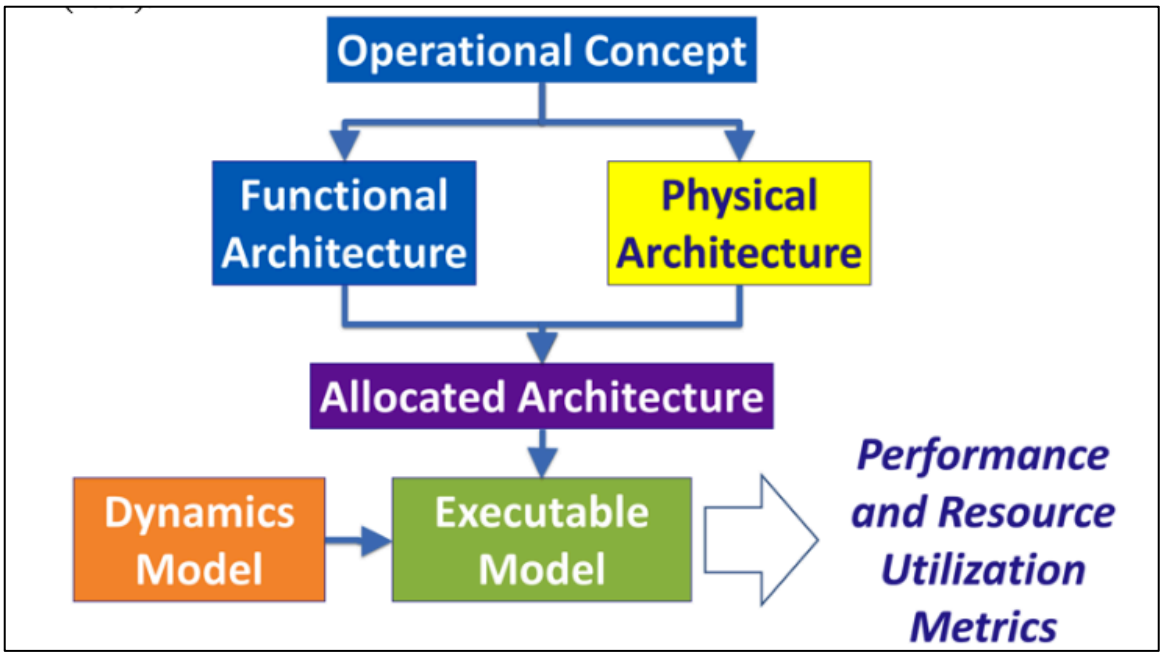


Figure 22. Reference Process for Synthesizing SoS Architectures.
Source: Kenley et al. (2014)

This paper is the most advanced consideration of SoS design with regard to MBSE and complete SoS architecting (including functional, physical, and allocated architectures). It is well linked with common MBSE products that makes using the process simpler when integrated with a larger MBSE systems engineering process. It is limited in that while it considers the one way relation of functional and physical architecting to allocated architecting, it does not allow for the reverse relationship. This impedes the development of a tradespace and the associated exploration of the trades among the functional, physical, and allocated architectures and SoS performance. It further does not explicitly account for the concept of participation risk or organizational architecture. The expansion and inclusion of this model into a greater SoS architecting and analysis analytic methodology would improve the state-of-the-art.

(4) Rao, Ramakrishnan, and Dagli, 2008

Rao et al. (2008) demonstrate a methodology to model the architecture of an SoS using SysML and then map that architecture to an executable Colored Petri Net (CPN) model. Using the Petri Net model, the researchers could assess the architecture according to their desired metrics. The demonstration used the Global Earth Observation System of Systems (GEOSS). The researchers used a methodology pictured in Figure 23. Note that the general flow is depicted on the bottom half: model the architecture in SysML, turn that into an executable model, and then use the executable model to evaluate the architecture. Though it is not expressly depicted, the authors note that following evaluation and analysis, changes can be made to the architecture and reassessed in an iterative manner.

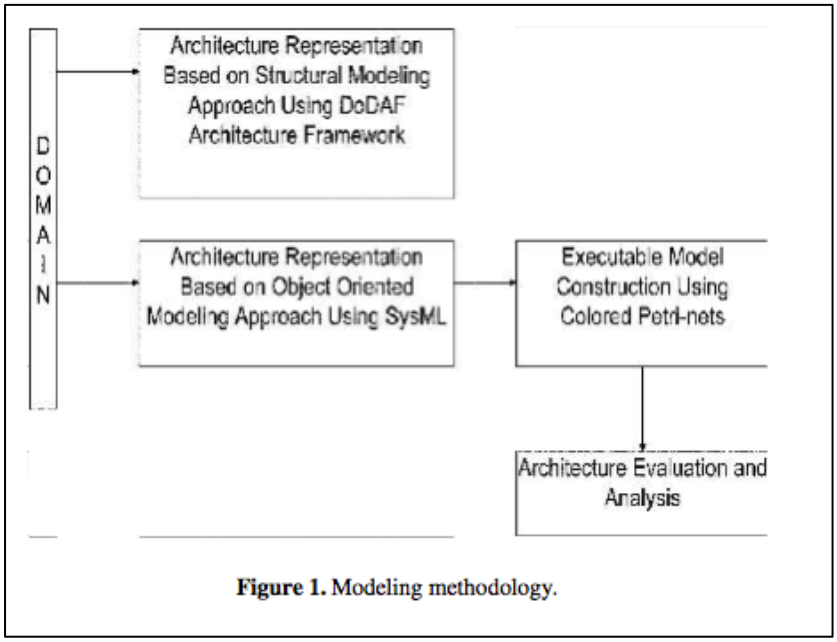


Figure 23. SysML and CPN Modeling Methodology. Source: Rao, Ramakrishnan, Dagli (2008)

This work provides a very concrete, useful manner in which to both model the architecture of a system and assess that architecture. It is limited in that the manner in which SysML allocates functions to components is static, which is at odds with a general

SoS dynamic allocated architecture. It is further limited in that Petri Nets can only model simple emergence. Finally, it is limited in that it does not expressly develop a tradespace or method for TSE; any iteration that occurs is, to an extent, a trial and error process which can be time consuming and ineffective for searching a large design space.

c. SoS Exploratory Design

The two pieces of literature that consider SoS tradespace exploration methods (Chattopadhyay 2009; Biltgen et al. 2006) are severely limited. Chattopadhyay (2009) presents a method for SoS TSE, but abstracts the challenge of defining SoS architectures to the problem of SoS composition and ignores other, significant considerations; furthermore, this work makes significant use of very low fidelity methods of defining system attributes that do not represent emergent properties. Biltgen et al. (2006) present an SoS TSE method, but the definition of an SoS is restricted to directed SoS with primarily physical architecture considerations. Neither work encompasses the requirement to consider different perspectives on SoS architecture and how that affects system performance.

(1) Chattopadhyay, 2009

Chattopadhyay (2009) presents, "A Method for Tradespace Exploration of Systems of Systems."[9] It is an extension to the "Dynamic Multi-Attribute Tradespace Exploration" (Ross 2006; Chattopadhyay 2009). The SoS Tradespace Exploration Method (SOSTEM) is seen in Figure 24. This is annotated as a ten step process:

1. "Determining the SoS Mission,"
2. "Generating a List of Component Systems,"
3. "Identifying Stakeholders and Decision-makers for SoS and Component Systems,"
4. "Classifying Component Systems According to Managerial Control and Participation Risk,"
5. "Defining SoS Attributes and Utility Information,"

[9] This work has been presented in various forms (Chattopadhyay et al. 2008; Chattopadhyay et al. 2009; Ross and Rhodes 2015) with no apparent material change to the research.

6. "Defining SoS Context Changes,"

7. "Modeling SoS Performance and Cost: a) Modeling Legacy Systems, b) Modeling New Systems, c) Modeling the SoS,"

8. "Tradespace Analysis,"

9. "Epoch-Era Analysis,"

10. "Selecting Value Robust SoS Designs" (Chattopadhyay 2009, 89).

This process yields an explorable tradespace that decision-makers may consider in designing an SoS.

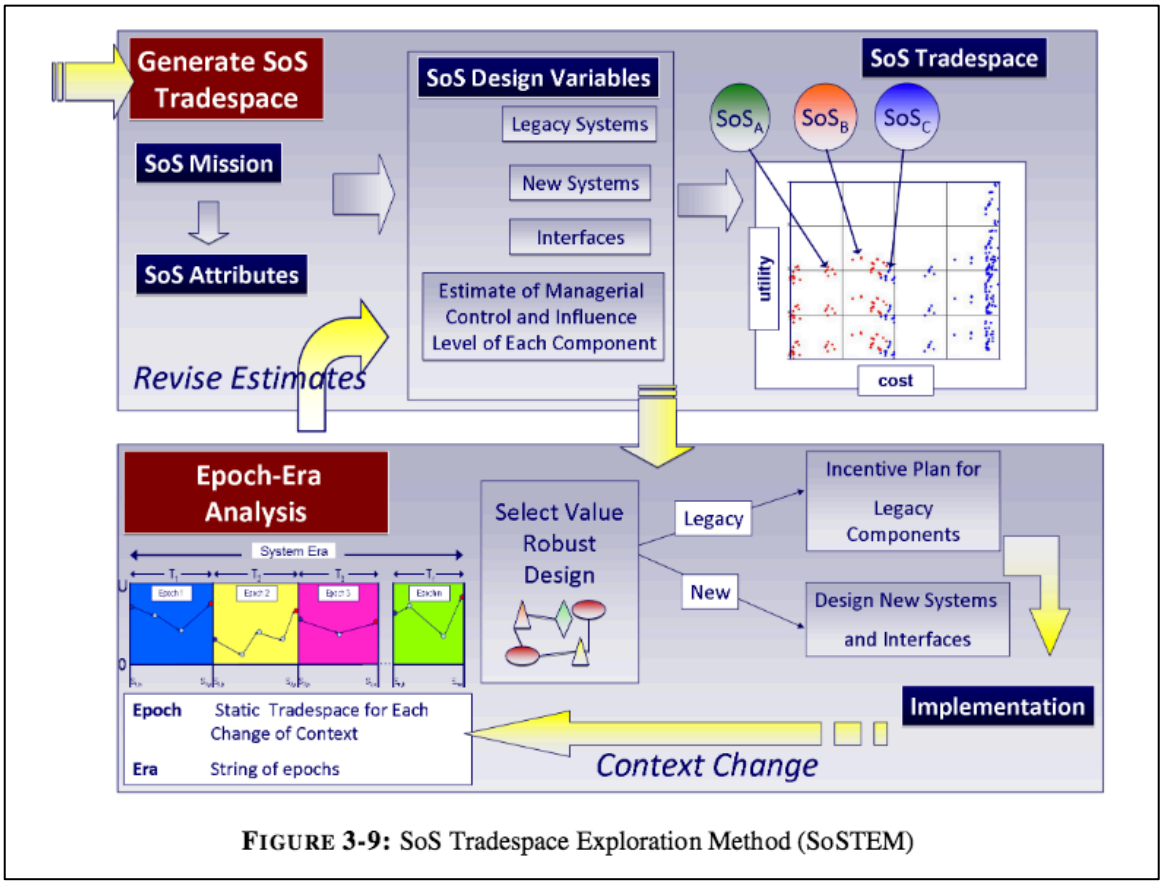


FIGURE 3-9: SoS Tradespace Exploration Method (SoSTEM)

Figure 24. SoS Tradespace Exploration Method.
Source: Chattopadhyay (2009)

Chattopadhyay's SOSTEM is the most useful of the current research on SoS design with regard to developing an explorable tradespace that incorporates the key

distinctions of SoS. Unfortunately, it makes many simplifying assumptions and is not embedded with common systems architecting products, nor does it lend itself to be embedded. With regard to architecting, the SOSTEM simplifies the architecting problem to a matter of SoS composition, although it does acknowledge that there is some additional work and cost required to make some systems interface properly. While simplifying assumptions must be made for all models, this is too simplistic for even high level conceptual SoS architecting. It is limited in its ability to explore varying physical, functional, or allocated architectures or non-material factors in SoS design. Furthermore, it assumes that participation risk on the part of any given system is static, and not a function of the SoS architecture, which is certainly not the case as the cost and benefit for participation in an SoS is clearly a function of the SoS architecture (e.g., Maier's heuristic regarding architecting to induce desired systems to participate). Finally, the method of SoS analysis is deceptively simple and highly limited as discussed in Section II.E.2.b. Despite these flaws, it is a useful baseline for advancing analytic tools to support SoS design.

(2) Biltgen, Ender, and Mavris, 2006

Biltgen, Ender, and Mavris (2006) developed a "hierarchical, surrogate modeling environment for SoS analysis" depicted in Figure 25. Their research problem was to develop a method for collaborative design and trade studies for simultaneous SoS and system level development. As depicted, the methodology integrates the MOEs and MOPs at each level through a top-down analysis. To mitigate problems of computational time and proprietary information, the researchers used parametric surrogate models of each system. Additionally, they developed neural network surrogates to model the interactions among the systems. Ultimately this yielded an explorable "universal tradeoff environment" that engineers could use to develop system and SoS level requirements for subsequent engineering.

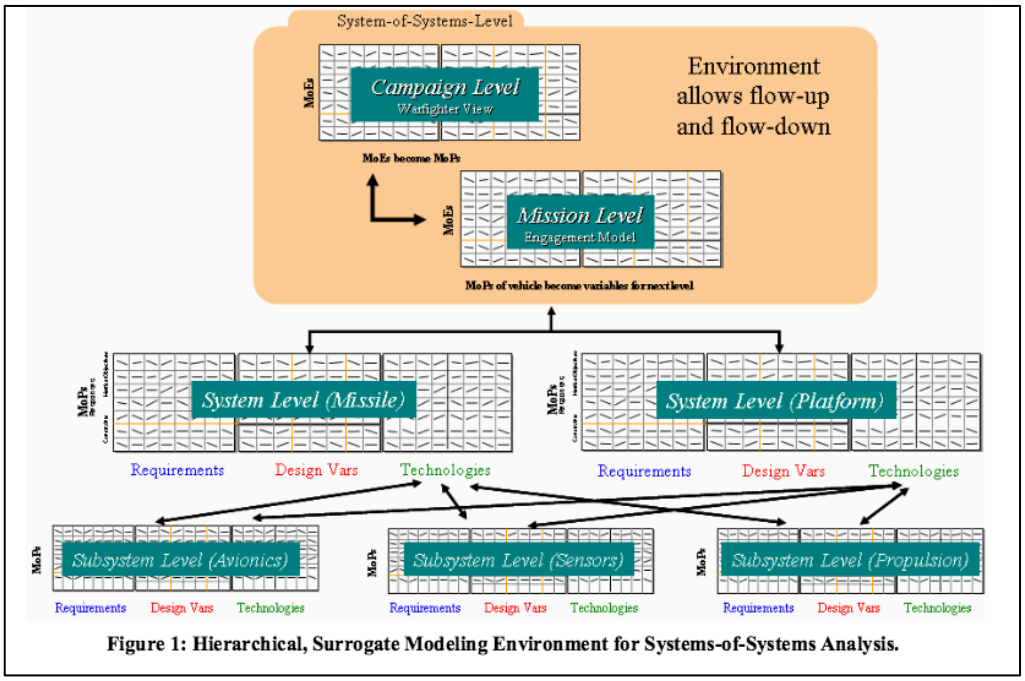


Figure 25. Hierarchical, Surrogate Modeling Environment for SoS Analysis. Source: Biltgen, Ender, Mavris (2006)

This work is useful as it clearly demonstrates a method of combining surrogate modeling for SoS analysis. Furthermore, it makes extensive use of data visualization and analysis to "illuminate the tradespace." It is limited to, what appears to be, directed SoS under development, though the authors do not explicitly state this. Furthermore, it makes no apparent use of commonly used systems architecting methods or provide methods of integrating these architecture models into the modeling and simulation methodology. Finally, it appears that the SoS architecture is relatively static in this methodology, and the exploration is more focused on how given an implicit functional, physical, and allocated SoS architecture the system level architectures and requirements are linked to SoS level MOEs and MOPs. This is a useful development, but only applicable to very unique SoSE problems, in particular, the development of a directed SoS from the bottom up.

F. CONCLUSION

SoS design is a challenging problem because designers must contend with pre-existing, independent (to varying levels) systems. Furthermore, SoS present emergent

behavior that is the product of interactions among various systems. A SoS is best represented through multiple perspectives—both technical and non-technical. One way to do this is to consider the physical, process, and organizational architectures of an SoS. By doing this, one is better able to assess an SoS design's potential operational performance through an ABM (or similar model). Unfortunately, by defining an SoS architecture in this manner, one significantly increases the size of the design space and explicitly defines the tradespace with parameters that cannot be assumed to be independent. This is a problem because current monolithic system TSE methods assume one can define design parameters in a manner such that they are independent or have limited interactions. On the other hand, current SoS design methods either do not account for the full requirements of an SoS architecture, or otherwise simplify the problem. Taken together, this creates a potential for an extension to the state-of-the-art of SoSE in the area of SoS TSE. The remaining chapters present these extensions.

III. THE SOS TRADESPACE DEFINITION METHODOLOGY THROUGH THE SOS ARCHITECTURE FEASIBILITY ASSESSMENT MODEL

This section introduces the primary contributions of this dissertation, the SoS Tradespace Definition Methodology (SoS-TDM) through the SoS Architecture Feasibility Assessment Model (SoS-AFAM). Together, these extend the state-of-the-art in two ways. Within MBSE, it extends the MBSE MEASA to be capable of addressing SoS and similar systems that must incorporate multiple, non-material factors in their architectures. Within SoSE, the SoS-TDM and the SoS-AFAM extend the state-of-the-art by augmenting current SoS design methodologies to include an exploratory design decision making method that considers multiple aspects of an SoS (physical, process, and organization) and by defining a general model for assessing SoS feasibility.

The SoS-TDM is predicated on the claim that, for any design space, the subset of that design space that contains the feasible design points is significantly smaller than the initial design space. Ultimately, it is impossible to prove this claim in complete generality; however, it is applicable in many (if not the majority) situations. Moreover, as a system increases in complexity,[10] it is generally more difficult to achieve a feasible design because there are more interactions among the sub-systems making it difficult for a system to meet all requirements. This only serves to further reduce the size of the feasible design space.

[10] The term "complexity" is used here generically. There are various technical definitions and measures of complexity; however, they do not serve the purposes here. A general concept of complexity may be considered as the number of interactions that occur among the sub-systems of a system.

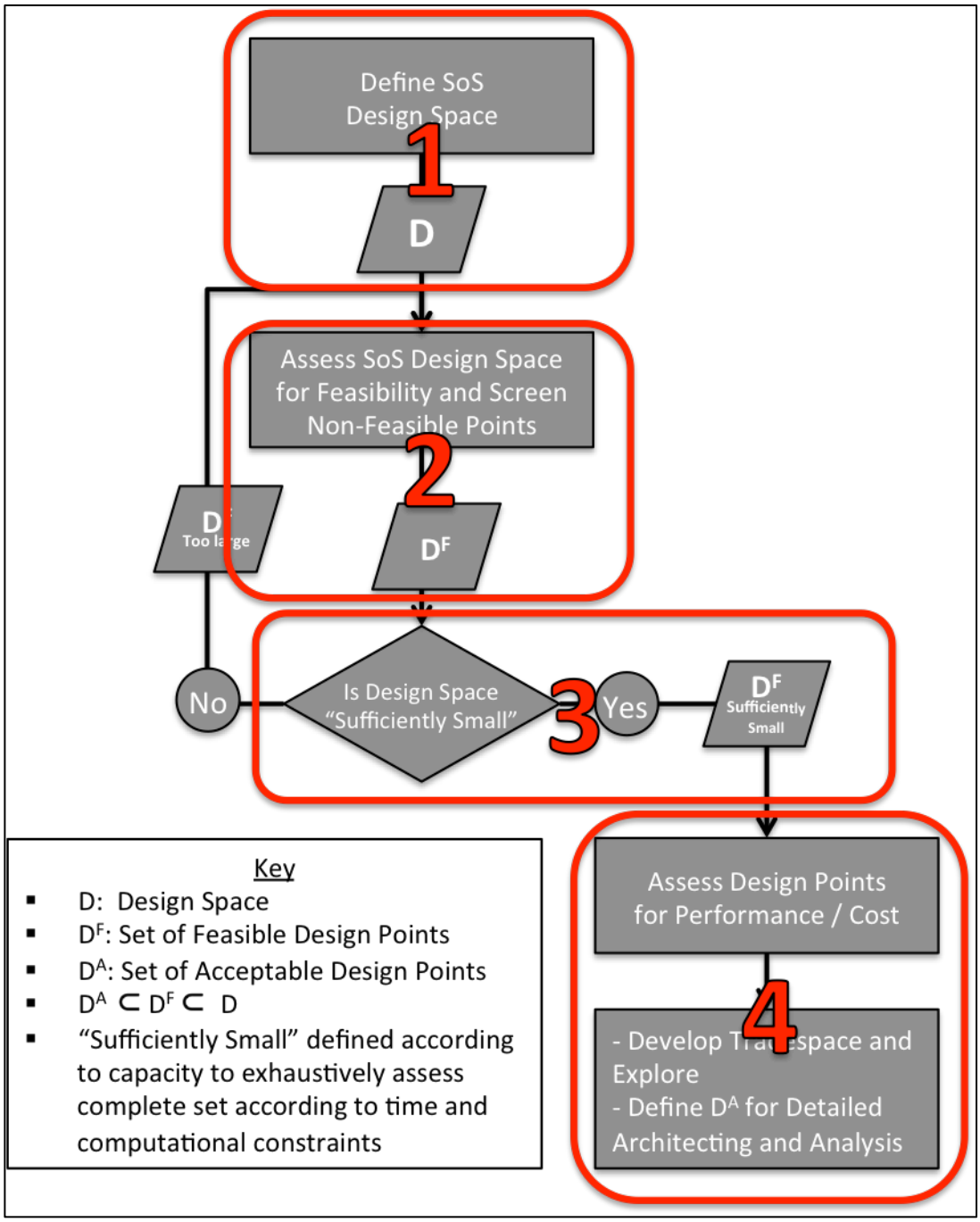


Figure 26. The SoS Tradespace Definition Methodology

The SoS-TDM is depicted in Figure 26. The methodology defines the tradespace of an SoS according to the parameters necessary for SoS architecting: physical, process, and organizational (Step 1). The feasibility model assesses the points in an SoS design space in an efficient manner to define the much smaller sub-set of the design space that is

feasible (Step 2). If the feasibility analysis winnows the design space sufficiently, one proceeds with design point analysis; otherwise, one iterates the first two steps (Step 3). These feasible design points may then be exhaustively analyzed for performance (Step 4). Taken together, the set of feasible design points and their associated performance attributes may form a tradespace that may be explored and inform subsequent detailed analysis.

A. SOS-TDM CONTEXT AND SCOPE

1. SoS-TDM in SoSE and MBSE

Within SoSE, the SoS-TDM occurs during the design phase(s) as depicted in Figure 27 and discussed in Section II.D.2. Areas of SoSE such as integration, test and evaluation, operations and maintenance are outside the scope of the SoS-TDM. Note that the SoS-TDM may be used in any choice of a general SoSE methodology, e.g., the iterated vee or wave models.

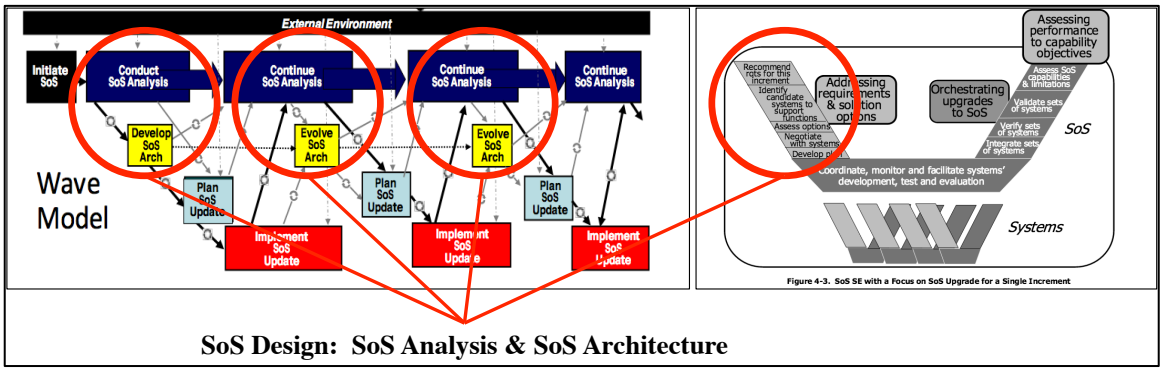


Figure 27. Where SOS-TDM is Useful in SoSE. Adapted from Dahmann et al. (2011) and Department of Defense (DOD) (2008)

In MBSE, the SoS-TDM facilitates design decision-making. In particular, the SoS-TDM is integrated with Beery's (2016) methodology, the MBSE MEASA. To solve the problem of not being able to define transfer functions between design parameters and operational parameters (as discussed in Section II.C.1.b) one re-orders the flow of the MEASA as depicted in Figure 28. In doing this, one defines the initial SoS requirements and top level functions similarly, but then uses that to inform the parameters necessary to

define the physical, process, and organizational architectures (the SoS design space), assesses this space for feasibility (what Beery calls synthesis) and then only assesses the feasible set of designs for operational performance and builds a tradespace.

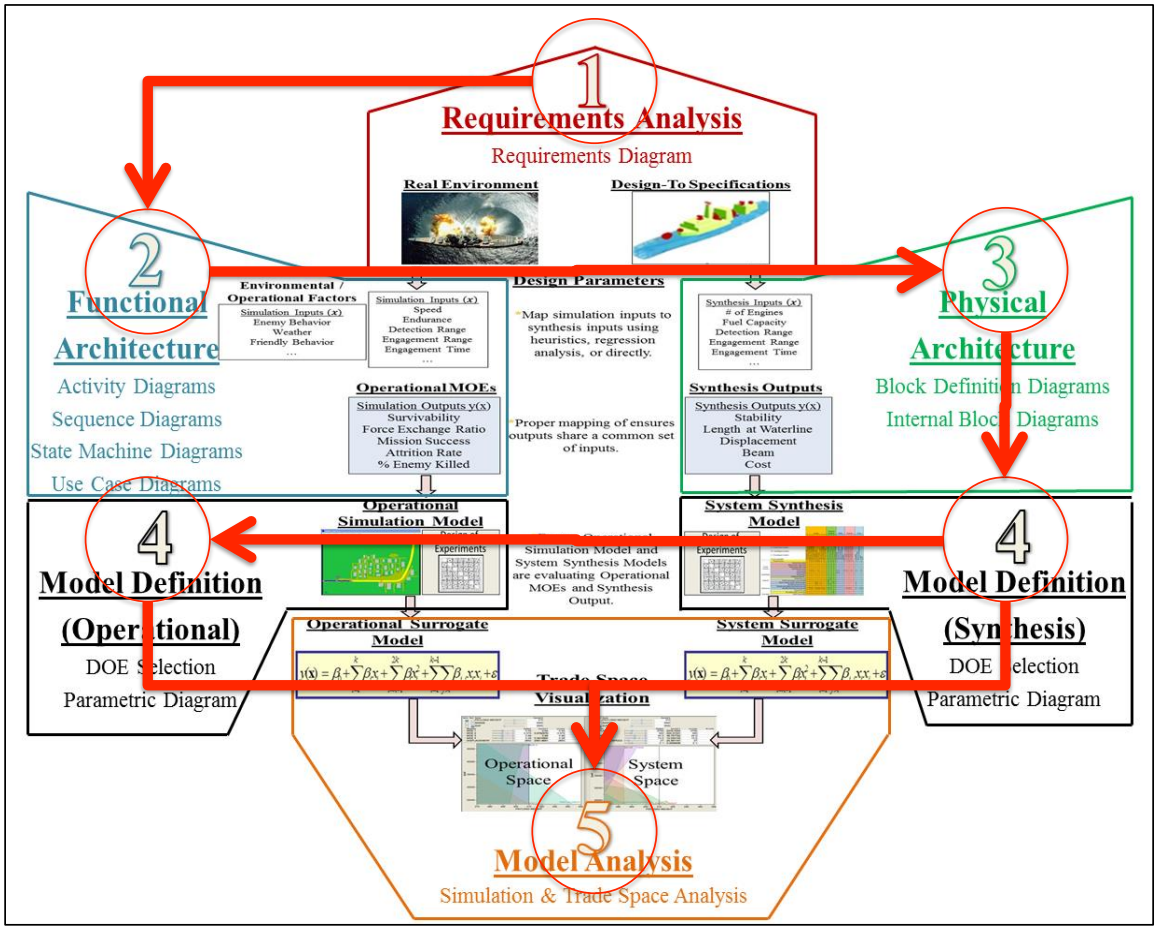
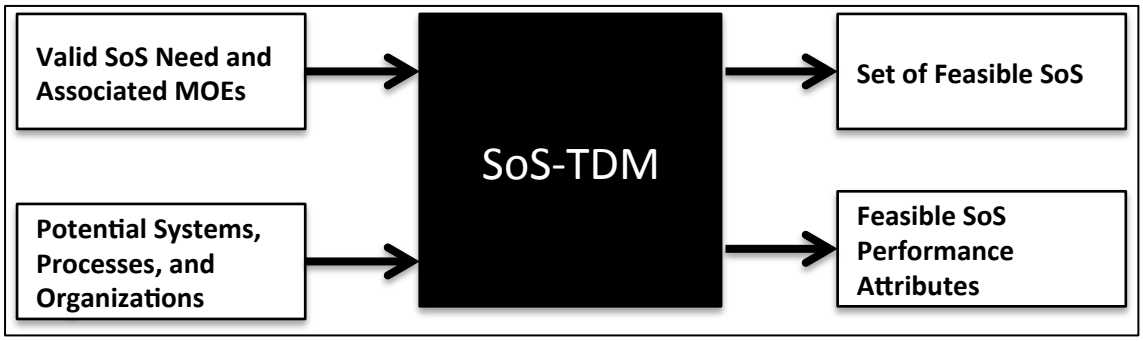


Figure 28. SoS-TDM Modification of the MBSE MEASA. Adapted from Beery (2016)

It is useful to consider the SoS-TDM as a system itself, and initially consider it a "black box" that takes inputs and produces outputs as seen in Figure 29. These inputs and outputs generally align with Beery's (2016) methodology, except where modified as necessary.



The figure shows the inputs and outputs of the SoS-TDM as a "black box" system in and of itself.

Figure 29. Inputs and Outputs of the SoS-TDM

The inputs include "Valid SoS Need and Associated MOEs" and "Potential Systems, Processes, and Organizations." These inputs are necessary for the SoS-TDM to build the set of possible SoS architectures that can meet the SoS need. The outputs include, "Set of Feasible SoS" and "Feasible SoS Performance Attributes." Together these two outputs define a tradespace, which may be explored by engineers and decision-makers.

The first input, "Valid SoS Need and Associated MOEs," is both a requirement and an underlying assumption. Foremost, the SOS-TDM requires a purpose against which to assess potential SoS. There must be some associated MOEs by which an engineer can 1) assess performance and 2) design the SoS to perform. Furthermore, this assumes, like the MEASA, that the engineer, analyst, and various stakeholders have developed a clear refined need that answers the stakeholders' problem(s) (Beery 2016). It does not assume that initial benchmarks for MOEs are completely valid, rather that they may be adjusted as one develops a better understanding of the tradespace.

The second input, "Potential Systems, Processes, and Organizations," is the list of possible systems that could be included in the SoS and the potential processes and organizations that may be used to govern the interactions among the systems to elicit the desired SoS emergent behavior(s).

The first output, "Set of Feasible SoS," is the set of SoS design points that are assessed as feasible by the SoS-AFAM. Each design point may be used to define an SoS

architecture that includes the necessary physical, process, and organization perspectives. Furthermore, these design points are directly linked to performance attributes as they form the inputs for performance models and simulations.

The second output, "Feasible SoS Performance Attributes," are the results of the models and simulations that are used to assess each feasible design point. The choice of model or simulation is dependent upon the desired MOEs and SoS need. These models are often ABM for operational considerations (e.g., percent collateral damage), but may also be deterministic (e.g., a cost model). The SoS-TDM and SoS-AFAM output a set of design points as inputs for an operational model. Typically, for an SoS, a reasonable operational model is an ABM with the agents representing the various systems. Importantly, the rules that govern an ABM – how agents interact, how agents make decisions, and so forth—are described by the design parameters of process and organization and required for use of the SoS-TDM and SoS-AFAM.

Together, the inputs and outputs define a tradespace for the SoS. This is a practical linkage between the synthesis model and the operational model as defined in the MBSE MEASA Step 4 (Beery 2016) and as seen in Figure 28. This may then be used as a part of a larger SoSE or MBSE process. The SoS-TDM and SoS-AFAM are tool and technique agnostic; they provide a methodology and framework for engineering problems that must be defined by parameters with significant interaction, i.e., SoS.

2. SoS-TDM Scope and Assumptions

The SoS-TDM is applicable to the design of acknowledged or directed SoS composed of pre-existing systems that produce desired emergent behavior(s) in a manner that may be understood and modeled. Each requirement for employment of the SoS-TDM is outlined as follows:

a. Type of SoS

The SoS-TDM is intended for use with acknowledged or directed SoS. These types of SoS have both a centrally agreed upon purpose and some level of a central administration or engineering (DOD 2008). The latter condition is a necessary

prerequisite for the use of the SoS-TDM. If an organization or person is using SoS-TDM to engineer an SoS, that SoS is, by definition, either acknowledged or directed. The former condition is necessary because the purpose of (and need for) the SoS is a major input of the SoS-TDM.

b. Type of Interfaces

The SoS-TDM assumes that the interfaces among the various constituent systems are purely information interfaces (i.e., communications sub-systems connect the various constituent systems). This is assumed for two reasons. First, generally speaking, SoS are of this form (Maier 1998). Second, information has the ability to be transformed across multiple communications systems with varying levels of efficacy. For example, information sent over a phone call from System 1 to System 2 may be transcribed and sent over email from System 2 to System 3. There may be a loss of information (e.g., the classic "Telephone Game"), but it is generally possible to do this. This is not the case, however, when one considers physical interactions. A piece of cargo of a certain size may be transferred over one physical cargo system (say in a freight train) but not in another physical cargo system (say an automobile). The case in which the systems of an SoS have physical interactions is therefore excluded from the SoS-TDM.

c. Pre-Existing Systems

The SoS-TDM only considers using pre-existing systems. This assumption allows the SoS-TDM to assume that these systems are well understood with meaningful, useful architectures and performance measures. This mitigates the SoS-TDM from having to vary the performance of individual systems within the SoS when assessing the SoS performance. The SoS-TDM does allow for a discrete number of re-factorizations of these systems. Again, the re-factorization is assumed to be well understood (e.g., adding an existing communications sub-system to a system that does not have that sub-system).

By assuming that the possible constituent systems pre-exist, the data that populates the analysis of the SoS is more accurate. This limits the number of assumptions one must make in developing the synthesis and operational models. This allows analysts to more clearly determine which variables are highly important and which are not.

It is reasonable to assume that an existing system does or can have a valid system architecture and valid models of its performance. Organizations maintain data on their systems and conduct operational test and evaluation routinely. This is a well-studied field with extensive practical experience. It is highly reasonable to assume that an existing system has well-developed data on its performance and mode of activity.

d. Predictable Systems

The final necessary assumption is that the constituent systems perform in some predictable manner. That is to say, for a given input to a system, it produces a predictable, if stochastic, output. A non-predictable system provides no regular output for a given input. This is a challenge, because systems that involve humans are not always predictable; however, humans, operating as a part of a system (say a military unit), can be expected to perform the standard procedures for that system and given situation. In the military, these are typically codified as tactics, techniques, and procedures. The analogous concept exists for other, non-military systems. This requirement allows the reasonable use of models of the system behavior.

B. SOS-TDM – DESIGN SPACE DEFINITION

The first step of the SoS-TDM is to define the design space for the SoS problem. This includes three things: the physical architecture design space, the process architecture design space, and the organization architecture design space. The SoS design space is the Cartesian product of these three sub-design spaces:

$$\boldsymbol{D = D^{Phys} \times D^{Proc} \times D^{Org}}$$

The set, $\boldsymbol{D}$, contains the eventual set of feasible SoS and, eventually, the set of acceptable SoS that will be chosen for detailed architecting and analysis. In the context of the greater SoS-TDM, this step is highlighted in red in Figure 30.

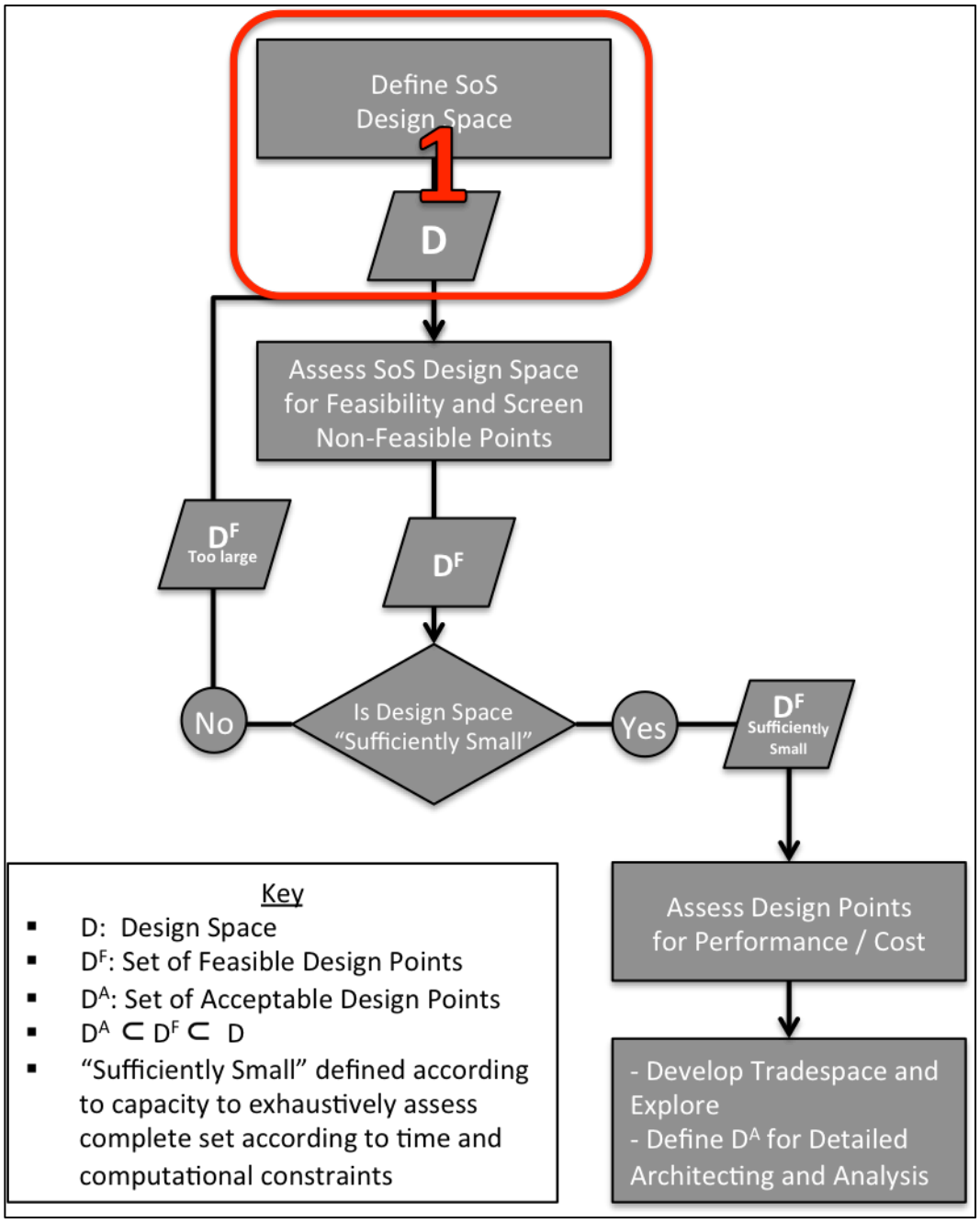


Figure 30. SOS-TDM – Define SoS Design Space

1. Physical Architecture Design Space

The physical architecture design space, $\boldsymbol{D^{Phys}}$, is the set of design points defined by the parameters that define the physical architecture. The physical architecture of a design point is the composition of the included constituent systems, system refactoring

parameters, and SoS bridges. Associated with each parameter, constituent system, refactoring, or bridge, are the various details of the parameter capabilities regarding communications and information flow. Together, these form a communications network topology. Mathematically the physical architecture design space may be defined as:

$$\boldsymbol{D^{Phys}} = \boldsymbol{S_1}\times\boldsymbol{S_2}\times \ldots\times\boldsymbol{S_i}\times \ldots\times\boldsymbol{S_n}$$

Each $\boldsymbol{S_i}$ may take a value in $< 0, 1, 2, 3, \ldots >$ where zero indicates that the i^{th} system is not included in the SoS, a one indicates that the system is included, and a value higher than one indicates that the system is included and refactored according to the specifications equated to that number.

For example, if one potential system in an SoS is a "U.S. Headquarters," called $\boldsymbol{S_4}$ as a parameter, it may take a value in $< 0, 1, 2 >$. If $\boldsymbol{S_4} = 0$, then it is not included in that SoS. If $\boldsymbol{S_4} = 1$ then the "U.S. Headquarters" is included in the SoS as is. If $\boldsymbol{S_4} = 2$, then that indicates the "U.S. Headquarters" is refactored to include an "Afghan Liaison."

As a special case, some systems may be included that exist solely as a bridge for the SoS. In this case, the parameter is still treated as a separate system, but this system exists solely for the purpose of serving as a bridge among the various constituent systems.

$|\boldsymbol{D^{Phys}}|$, denotes the size of $\boldsymbol{D^{Phys}}$ and is the product of the magnitude of each parameter's domain, call this number s_i or b_j as it corresponds to each parameter. Using the previous example, $\boldsymbol{S_4} \in < 0, 1, 2 >$ has a magnitude of three, so $s_4 = 3$. In general, each system or bridge may take at least two values, inclusion or exclusion. Therefore,

$$\left|\boldsymbol{D^{Phys}}\right| = s_1 \cdot s_2 \cdot \ldots \cdot s_n \geq 2^n$$

$\left|\boldsymbol{D^{Phys}}\right| \leq M_s^n$, where M_s is the maximum of all s_i

Each design point in $\boldsymbol{D^{Phys}}$, coupled with the corresponding system, refactorization, and bridge data**,** may define a unique physical architecture. The analysis of each physical architecture is discussed in Section III.C.1.

2. Process Architecture Design Space

The process architecture design space is the set of design points defined by the parameters that define a process architecture. The process architecture of an SoS defines the sequence of operational activities and operational rules. Mathematically, this may be defined as:

$$\boldsymbol{D^{Proc} = F_1 \times F_2 \times \ldots \times F_i \times \ldots \times F_a \times E_1 \times E_2 \times \ldots \times E_j \times \ldots \times E_b}$$

Each $\boldsymbol{F_i}$ is a set of mutually exclusive operational activity sequences (i.e., one must pick one and only one of the sequences available from that set). Each operational activity sequence may be assigned a nominal value (e.g., sequence 1 or 2) to define a design point. If there are multiple operational activity sequences that are not mutually exclusive (e.g., the SoS performs different sequences of activities to produce different desired emergent behaviors), these are represented by distinct F_i. The total number of sets of operational activity sequences is *a*. For example, in an indirect fire scenario, we define two, mutually exclusive operational activity sequences:

1. Observe → Shoot
2. Observe → Deconflict → Shoot

Thus, in this example, there is one F_i:

$$F_1 = <\text{“Observe} \rightarrow \text{Shoot,” “Observe} \rightarrow \text{Deconflict} \rightarrow \text{Shoot”}>,$$

which may be shortened as

$$F_1 = <1, 2>.$$

Each $\boldsymbol{E_j}$ indicates if the j^{th} employment rule is used. An employment rule (or rule of employment) is a rule that prescribes how systems within the SoS must behave. There are *b* sets of rules of employment. For example, in an indirect fire scenario, we define two distinct rules of employment, one concerning the number of required observations of a target prior to shooting (1 or 2) and another concerning the rules of engagement for shooting at targets near civilians (authorized or not). Thus, in this example, $b=2$ and E_1 and E_2 are defined as:

$$E_1 = <\text{“One required observation,” “Two required observations”}>$$

E_2 = <*"May shoot near civilians," "May not shoot near civilians"*>

Together, F_1, E_1, and E_2 define a process architecture design space by their Cartesian product.

$|\boldsymbol{D^{Proc}}|$ denotes the size of $\boldsymbol{D^{Proc}}$ and is the product of the magnitude of the domain of each parameter, f_k *or* e_l. In general, each parameter has at least two potential values, therefore

$$|\boldsymbol{D^{Proc}}| = (\Pi_{k=1}^{a} f_k) \cdot (\Pi_{l=1}^{b} e_l) \geq 2^{a+b}$$

$|\boldsymbol{D^{Proc}}| \leq M_f^a \cdot M_e^b$, where M_f and M_e are the maxima of f_k and e_l

Each design point in $\boldsymbol{D^{Proc}}$ coupled with the associated values for the parameters defines the process architecture for that SoS. The analysis of a process architecture is discussed in Section III.C.2

3. Organizational Architecture Design Space

The organizational architecture design space is the set of design points whose parameters describe the organizational architecture of the SoS. The organizational architecture describes the relationship between each pair of systems within the SoS. This may be described mathematically as

$$\boldsymbol{D^{Org}} = \boldsymbol{I_{12}} \times \boldsymbol{I_{13}} \times \ldots \times \boldsymbol{I_{ij}} \times \ldots \times \boldsymbol{I_{(n-1)n}}$$

Each $\boldsymbol{I_{ij}}$ takes a value that corresponds to a predefined relationship between two systems. Note that there is no parameter for $\boldsymbol{I_{ii}}$; that is, there is no defined relationship for a system with itself. For example, if there are two systems, a U.S. Headquarters, $\boldsymbol{S_4}$, and a Special Operations Forces Team, $\boldsymbol{S_5}$, and there are four defined relationships: no relationship, a collaborative relationship, and a command-subordinate relationship, we may define

$$\boldsymbol{I_{45}} \in <' \, No\ Relationship', ' \, CollaborativeRelationship', ' \, Commander', 'Subordinate' >$$

and

$$\boldsymbol{I_{54}} \in <' \, No\ Relationship', ' \, CollaborativeRelationship', ' \, Commander', 'Subordinate' >$$

The size of $\boldsymbol{D^{Org}}$, $\left|\boldsymbol{D^{Org}}\right|$, is the product of the magnitude of the domain of each $\boldsymbol{I_{ij}}$, call this i_{ij} Note that, at a minimum, there are always two organizational relationships: no relationship or some other relationship. Thus, the magnitude of $\boldsymbol{D^{Org}}$ is:

$$\left|\boldsymbol{D^{Org}}\right| = i_{12} \cdot i_{13} \cdot \ldots \cdot i_{ij} \cdot \ldots \cdot i_{n(n-1)} \geq 2^{n(n-1)}$$

$\left|\boldsymbol{D^{Org}}\right| \leq M^{n(n-1)}$, where M is the max of all i_{ij}

In reality, defining the organizational design space in this combinatorial manner is untenable. The design space becomes enormously large for SoS with more than four potential systems. For example, with four possible relationships and nine possible systems, using a combinatorial approach leads to an organizational design space with $4^{9 \cdot 8} \approx 2.2 \times 10^{43}$ distinct design points. To resolve this issue, one may define a number of distinct organizational architectures heuristically. Each of these may be defined using some set of well-defined relationships and a matrix whose *i-j* entries correspond to the relationship between the i^{th} and j^{th} systems. In this manner, we may define $\boldsymbol{D^{Org}}$ explicitly as <*Organization 1, Organization 2, ... Organization o*>, where *o* is the number of defined organizations. Accordingly: $\left|\boldsymbol{D^{Org}}\right| = o \geq 2$

Each point in $\boldsymbol{D^{Org}}$ coupled with the defined relationships defines an organizational architecture for the SoS. This closely mirrors what is represented by the OV-4: Organizational Relationships view in DODAF (DOD CIO 2010). Examples of pre-defined relationships include operational control (OPCON), tactical control (TACON), Direct Support (DS), General Support (GS), administrative control (ACON), coordinating authority, and direct liaison authorized (DIRLAUTH) (Joint Chiefs of Staff [JCS] 2011). For non-DOD SoS, one must carefully define these relationships according to information requirements and the affects a relationship has on system decision-making.

The analysis of the organizational architecture is discussed in Section III.C.3. In defining an organizational design point in $\boldsymbol{D^{Org}}$ an engineer is advised to consider this wealth of literature and any pre-existing organizational relationship definitions or requirements. This facilitates subsequent integration activities.

4. SoS Design Space

The design space for an SoS is defined as the Cartesian product of the physical, process, and organization architecture design spaces as defined in the previous three sections. This design space has a magnitude:

$$|\boldsymbol{D}| = \left|\boldsymbol{D^{Phys}}\right| \cdot |\boldsymbol{D^{Proc}}| \cdot \left|\boldsymbol{D^{Org}}\right| \geq 2^{n+a+b} \cdot o$$

For values of *n, a,* and *b* such that their sum is greater than or equal to 16, the magnitude of the design space is non-trivial (greater than 100,000), and increases rapidly on the order of 2^n. Direct assessment of each design point in the total design space is either impractical or impossible. The SoS-TDM contends with this issue through feasibility analysis of potential design points.

C. SOS-TDM – DESIGN SPACE FEASIBILITY ANALYSIS AND SCREENING: THE SOS-AFAM

The second step of the SoS-TDM is the design space feasibility analysis and screening as depicted in Figure 31. The goal of this step is to define a set of feasible SoS "sufficiently small" so that each design point can be evaluated. This yields the feasible design space, $\boldsymbol{D^F}$:

$$\boldsymbol{D^F} =< \boldsymbol{d} \in \boldsymbol{D} | f_{feasible}(\boldsymbol{d}) = 1 >$$

The function that assesses an SoS design point for feasibility is called:

$$f_{feasible}: \boldsymbol{D} \rightarrow [0, 1]$$

where a design point is feasible if it returns a value of one and infeasible if it returns a value of zero. The challenge is to define a $f_{feasible}$ that is accurate, computationally efficient, and practical.

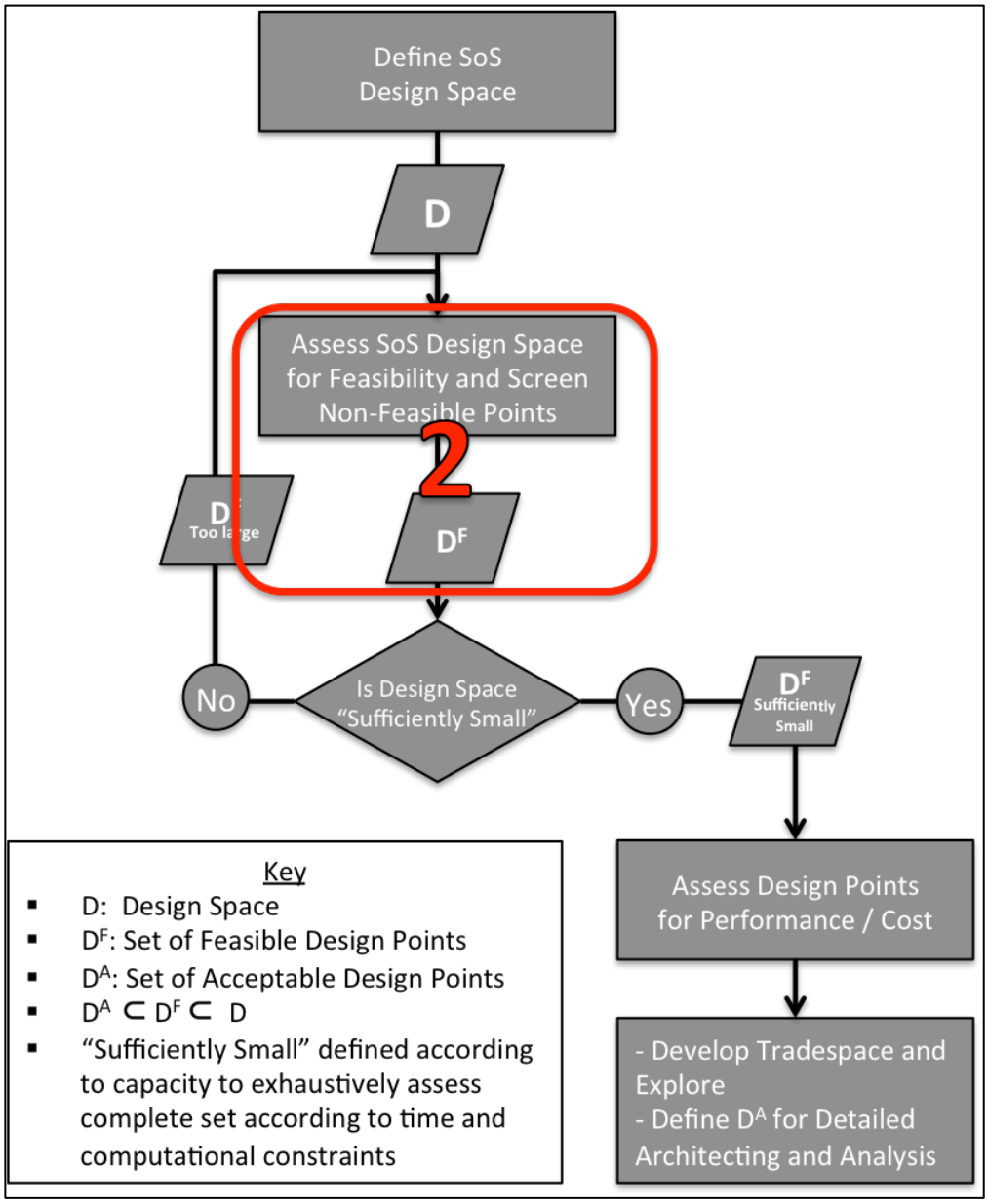


Figure 31. SoS-TDM – Design Space Feasibility Analysis and Screening

The SoS-TDM – Design Space Analysis is accomplished through the SoS-AFAM. This defines the SoS feasibility function through multiple steps that analyze subsets of the design space—the physical, process, and organizational—individually and then together as depicted in Figure 32. The steps of the SoS-AFAM are listed and correspond with the numbers in the figure:

1. Physical Design Space Feasibility Analysis: $f_{phys}: \boldsymbol{D^{Phys}} \rightarrow [0, 1]$

2. Process Design Space Feasibility Analysis: $f_{proc}: \boldsymbol{D^{Phys} \times D^{Proc}} \rightarrow [0, 1]$

3. Organization Design Space Feasibility Analysis: $f_{org}: \boldsymbol{D^{Phys} \times D^{Org}} \rightarrow [0, 1]$

4. Total Design Space Feasibility Analysis: $f_{feasible}: \boldsymbol{D} \rightarrow [0, 1]$

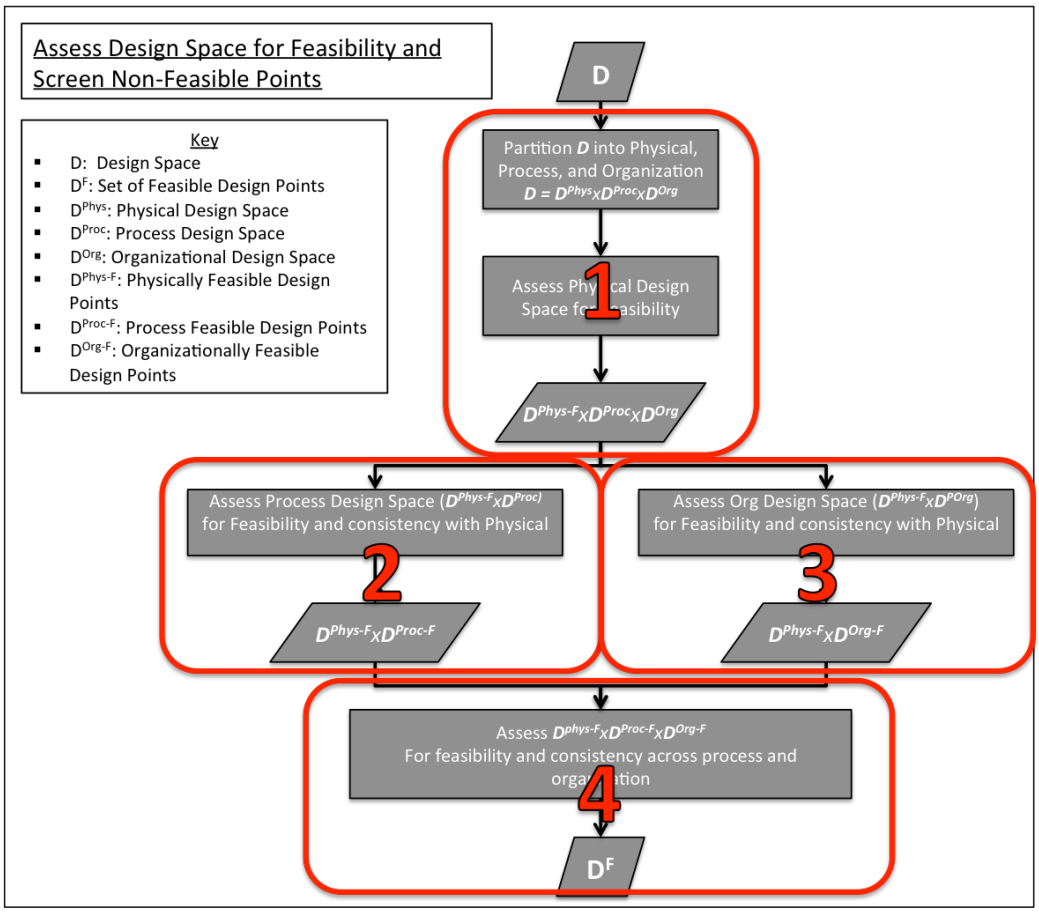


Figure 32. The SoS-AFAM

Each function is employed through a standard flow seen in Figure 32. Each function is implemented through a computer algorithm that checks each SoS design point against a minimum set of requirements that are defined as necessary, but not necessarily sufficient, for any SoS architecture (e.g., the network topology of the included systems must be connected). The output of each function is then assessed against the next function until the entire design space has been assessed. The initial screen is a high-level, low-fidelity analysis. One can then

iterate through increasing levels of fidelity for SoS feasibility until one defines a feasible subset of the design space that is "sufficiently small."

One significant advantage of this methodology is that it partitions the design space into sub-spaces that are progressively screened for feasibility, thus reducing the requirement to check every point in the design space. For example, if there is a physical design point, $\boldsymbol{d^{phys}} \in \boldsymbol{D^{Phys}}$, there are many design points in the overall in design space that include this physical design as a part of them. Specifically, there are $|\boldsymbol{D^{Proc}}| \cdot |\boldsymbol{D^{Org}}|$ design points in $\boldsymbol{D}$ that have the same physical parameters as $\boldsymbol{d^{Phys}}$. Through one calculation, if we assess $\boldsymbol{d^{Phys}}$ as infeasible, then every point in the overall SoS design space with those parameters is also infeasible and may be eliminated without further analysis.

1. Physical Design Space Feasibility Analysis

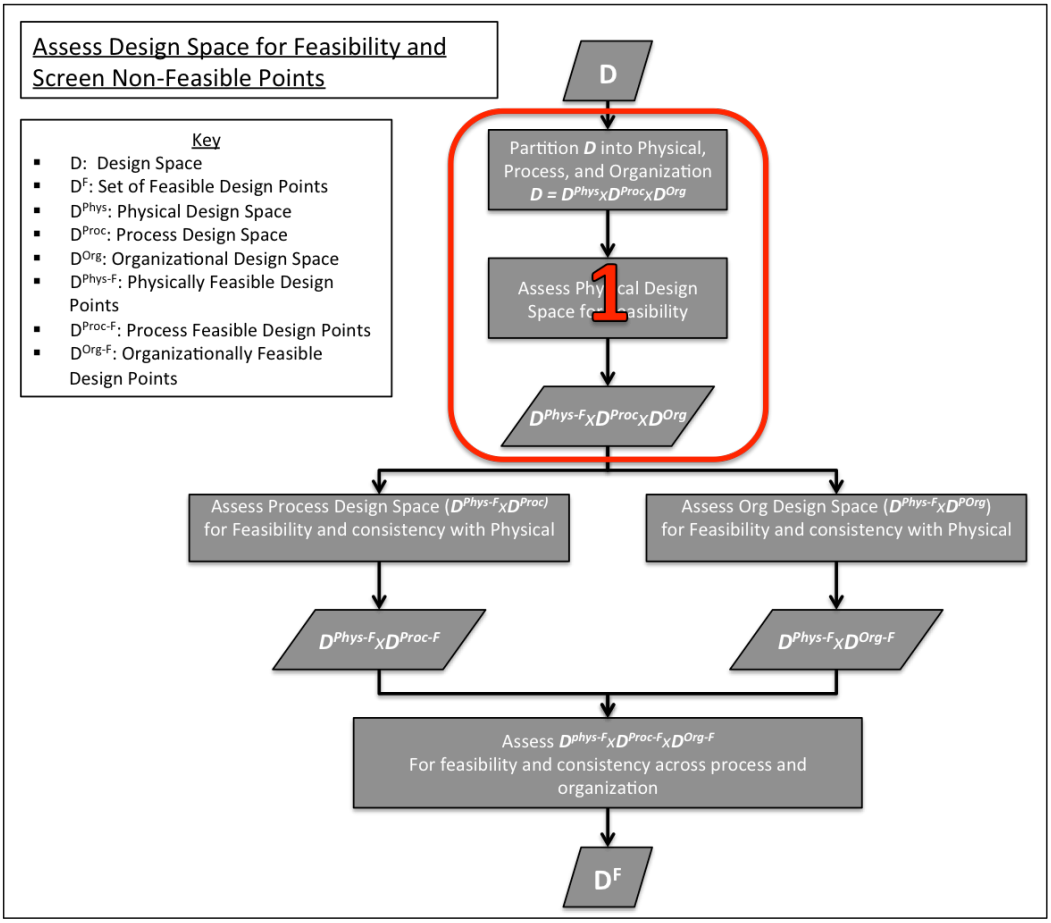


Figure 33. SoS-AFAM Step 1: Physical Design Space Feasibility Analysis

Figure 33 depicts the first step of the SoS-AFAM, the physical design space feasibility analysis. In this step, potential SoS designs are assessed for physical feasibility. This test only assesses the design points in $\boldsymbol{D^{Phys}}$, not the entire design space.

This test may take varying levels of fidelity, but ultimately rests upon the idea that an SoS is only feasible (from a physical perspective) if every system is connected to every other system, either directly or indirectly. That is, if the physical composition of the SoS coupled with its communication interfaces forms a connected network, the SoS is feasible.

A connected network is one in which every node (in this case system) can form a path to every other node (system). A path is a set of nodes and their edges (in this case communications interfaces) that form a continuous string from one node to another (Newman 2010). Figure 34 shows examples of connected networks, paths, and non-connected networks. It is intuitively clear that, for an SoS to function and include all of its constituent systems, it must form a connected network. In the lower left-hand quadrant of the figure, node A is not connected to the network. Were A, B, C, and D an SoS, System A would have no means of communicating with the other systems. It would make no difference if A were there or not to systems B, C, and D.

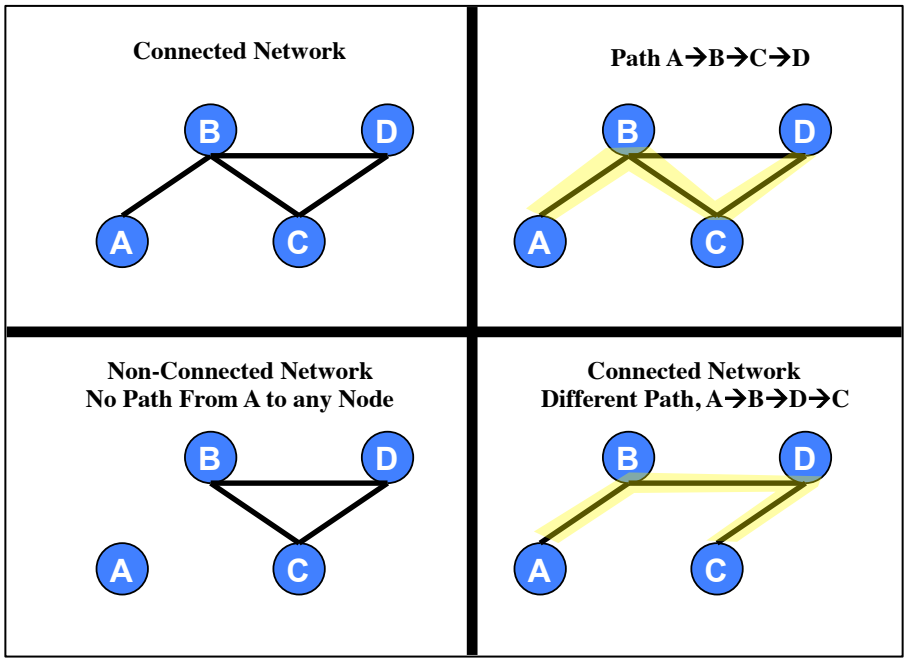


Figure 34. Examples of Connected Networks and Paths

a. Initial Physical Feasibility Test

The initial physical feasibility test requires two inputs: the included systems (and their re-factorizations) and a table of the available means of communication to each system (or re-factorization). The data comes from the definition of each design point and is in the form

$$< S_1, S_2, \ldots, S_n >$$

where each parameter takes a value from a set of pre-defined possible values, including 0 for exclusion of the system, 1 for inclusion of the system as is, and 2 or greater for a re-factorization of the system. The table of possible systems versus communications means comes from the system data. This is pre-defined, as the systems already exist. In DODAF, this data would come from a system's DIV-1, 2, or 3, SV-1, 3, or 6 (DOD CIO 2010). For non-DOD systems, this data may come from a similar format, or may require an engineer to create it.

A simple example of a system versus communication table is seen in Table 3. This table outlines a theoretical set of possible systems (the names and details of this are discussed in greater detail in Chapter IV) and their communications sub-systems. This example assumes that if two systems have a mutual communications system, that they may communicate. An X in the *i-j* cell indicates that the i^{th} system may communicate with the j^{th} system. The X* indicates that a communication system is available if a re-factorization is employed.

Table 3. System versus Communication Type Table

	Afghan Artillery	**U.S. Artillery**	**Afghan HQ**	**U.S. HQ**	**U.S. PLT**	**SF Team**	**Afghan PLT 1**	**Afghan PLT 2**	**UAV**
Afghan FM	X		X	X*		X	X	X	
OSRTV				X					X
U.S. FM		X		X	X	X			
BFT		X		X	X	X			
MIRC		X		X					X

Using the system and communications table, one may form an adjacency matrix[11] that describes the network topology of a given set of systems and then assess it for connectedness. This may be done using the following algorithm:

```
Algorithm 1.   Physical Feasibility Initial Algorithm
COMMENT: Define the list of potential physical SoS
     compositions
     FOR each value of S_1
          FOR each value of S_2
               (Define a FOR loop for each S_i)
                    FOR each value of S_n
                         Define a vector [S_1, S_2, … S_n]
                              IF  Number  of  non-zero
                                  elements in the vector
                                  is greater than or equal
                                  to two
                              Add  vector  to  list  of
                                  potential physical SoS
                                  compositions
                         ENDIF
                    ENDFOR S_n
               …
          ENDFOR S_2
     ENDFOR S_1
COMMENT: End Define the list of potential physical SoS
     compositions

COMMENT: Assess potential physical SoS compositions for
     connectedness
FOR each potential physical SoS composition
     DEFINE a square matrix of zeros of with the size of
          the number of included systems (adjacency matrix)
     FOR each included system (i)
          FOR each included system (j)
               IF i and j are distinct AND the ith and jth
                    system share a common communications
                    system
```

[11] An adjacency matrix is a matrix whose entries correspond to the relationship of the respective row and column (Newman 2010).

```
                Enter a 1 in the ij entry of the adjacency
                     matrix
                ENDIF
           ENDFOR
      ENDFOR
      CALL FUNCTION "ISCONNECTED"[12] and assess if the
           adjacency matrix is connected.
      IF the adjacency matrix is connected
           ADD physical SoS composition to physically
           feasible list
      ENDIF
ENDFOR each potential physical SoS composition
```

This algorithm outputs the set of physical SoS compositions that meet the minimum requirement to form a connected network, shared common communications. Potential SoS compositions must meet this basic level of connectivity to meet any higher fidelity assessments of SoS connectivity. Call the resultant physical SoS design space:

$$\boldsymbol{D^{Phys-F}} = < \boldsymbol{d} \in \boldsymbol{D^{Phys}} | \boldsymbol{d}\ assessed\ as\ feasible\ by\ Initial\ Physical\ Feasibility\ Test >$$

b. Expanded Physical Feasibility Tests

If desired or necessary to further prune the design space, one may define progressively stricter physical connectivity tests upon a composition of systems. These tests may take a variety of forms and should be used as appropriate. These include range, system availability, minimum bandwidth (across the network), maximum latency, and maximum error rate.

The first expanded test is a simple refinement on the initial test. It assesses the distance between two systems and modifies the communications matrix if the distance between two systems exceeds the maximum range of a communications sub-system. This requires the knowledge of each system's location, or a reasonable approximation of its average location and the maximum range of each communications sub-system. One

[12] Network connectedness algorithms are well documented (Ahuja et al. 1993; Newman 2010) and readily available in a variety of network science packages for many programming and scripting languages.

calculates the distance between two systems in the most appropriate manner[13] and compares this distance two the maximum range of each sub-system shared between those two subsystems. One then assesses for connectivity in accordance with the same algorithm, as detailed in Section III.C.1.a.

Another connectivity test may measure the general availability of both the systems and communications sub-systems. Each system may have an operational availability, A_0 defined for it. One can then use this to simulate the connectivity of the SoS network when systems or communications sub-systems are unavailable. This is done by assessing a system's inclusion or not based upon its A_0 and then assessing the system connectivity in the same manner as Section III.C.1.a. This may be repeated an appropriate number of times (i.e., 30 or more) to give a percentage likelihood of connectivity. A more refined method of doing this would be to use the mean time between failure (MTBF) and mean time to recovery (MTTR) for each system and sub-system and conduct a simulation over a relevant time period that induces failures and recovery times on various systems according to the MTBF and MTTR and then assessing the percent time of connectivity. Decision-makers may then establish a minimum threshold as desired.

The next three measures assess different aspects of network connectivity. They are: minimum allowable bandwidth between any two systems in the network, maximal allowable latency between any two systems, and maximum allowable error rate. These tests may be done using the common, precise measures in terms of bits per second, seconds between transmissions, or percent corrupted bits respectively. Alternatively one

[13] This method may vary depending upon the distance between the systems and the type of communications sub-system. If the distance is relatively small, standard Euclidean distance measures in two or three dimensions are appropriate. If the distance is large (say with satellites), one may need to employ a different metric. Furthermore, if the communication sub-system is supported by relays (e.g., a cell or satellite telephone), the question may be the distance between each system and its nearest relay. In other cases, such as two systems having internet access, the distance may be irrelevant depending on exact communications requirements (e.g., latency, bandwidth).

may make a lower fidelity approximation if necessary.[14] To do this, one must define these measures on each system and communications sub-system.

One may measure the minimum bandwidth between any two systems in an SoS by considering the bandwidth of each communications sub-system. In general, the minimum bandwidth transmission between any two systems in an SoS is the minimum of the maximum bandwidth available to any system in the SoS. A decision-maker may determine a system infeasible if its minimum bandwidth does not exceed a certain threshold.

One may assess the minimum latency between any two systems in an SoS by defining a network flows problem where a network is defined for the SoS where each there is a node for each system and its communication type (e.g., if the "UAV" has "OSRTV" and "MIRC" communications sub-systems, there is a "UAV-OSRTV" node and a "UAV-MIRC" node. One then defines a link between each node that shares a common communications sub-system (e.g., there is a link between "UAV-OSRTV" and "U.S. HQ – OSRTV" as both the "UAV" and "U.S. Headquarters" share the common "OSRTV" communications sub-system), and weight that link with the latency of the communications sub-system. Furthermore, for any system that has multiple communications sub-systems, one defines the latency between those two nodes as the time it takes to reconfigure the information from one type of communications sub-system to another (e.g., there would be a link between "UAV-OSRTV" and "UAV-MIRC" weighted with the length of time it would take a system to reconfigure the OSRTV information into MIRC information). If a system cannot reconfigure information from one type into another, that link has a weight of zero. This results in a weighted adjacency matrix of size $C_1 + C_2 + \cdots C_i + \cdots C_n \leq m \cdot n$, where C_i is the number of communications sub-systems the i^{th} system has, m is the total number of communications sub-systems available and n is the number of systems available. To find the shortest path

[14] Those measures are highly useful when dealing with digital transmissions, e.g., e-mail. The concepts may be the same, but the actual measures less useful when considering non-digital communications, e.g., FM radio. One may consider the time it takes to send a standard formatted message in cases like this, e.g., a the Army standard for a Call For Fire should be made in 30 seconds or less.

between any two systems one considers each pair of systems and uses a shortest path algorithm to determine the shortest path between the two nodes in terms of latency.[15] There are multiple variations on Dijkstra's algorithm to assess shortest path for an adjacency matrix with non-negative weights (Ahuja et al., 1993) that have been codified in a variety of network science packages. Thus, with the formulation of the adjacency matrix as described, one can solve this problem for each pair, and define the maximal minimum latency between any two systems in the SoS.

The final test is the maximal allowable error rate. This problem is very similar to the previous problem. One must define the error rate for each communications sub-system and for each system's internal transmission between its own sub-systems. One then assesses the minimum error rate between two systems as the "shortest path" between the two systems along this error rate adjacency matrix. One can then assign a maximal allowable error rate for feasibility and eliminate systems that do not achieve this.

Note that these tests should only be used after the initial iteration of feasibility tests and winnowing of infeasible solutions. Depending upon the size and density of the network formed by an SoS, some of these tests could potentially take significant time. As with all models and methods, an engineer must take care to define the problem well for the given situation.

[15] Note that this does not require checking the shortest path between every member of the adjacency matrix as defined, as each system may have multiple instantiations in this adjacency matrix.

2. Process Design Space Feasibility Analysis

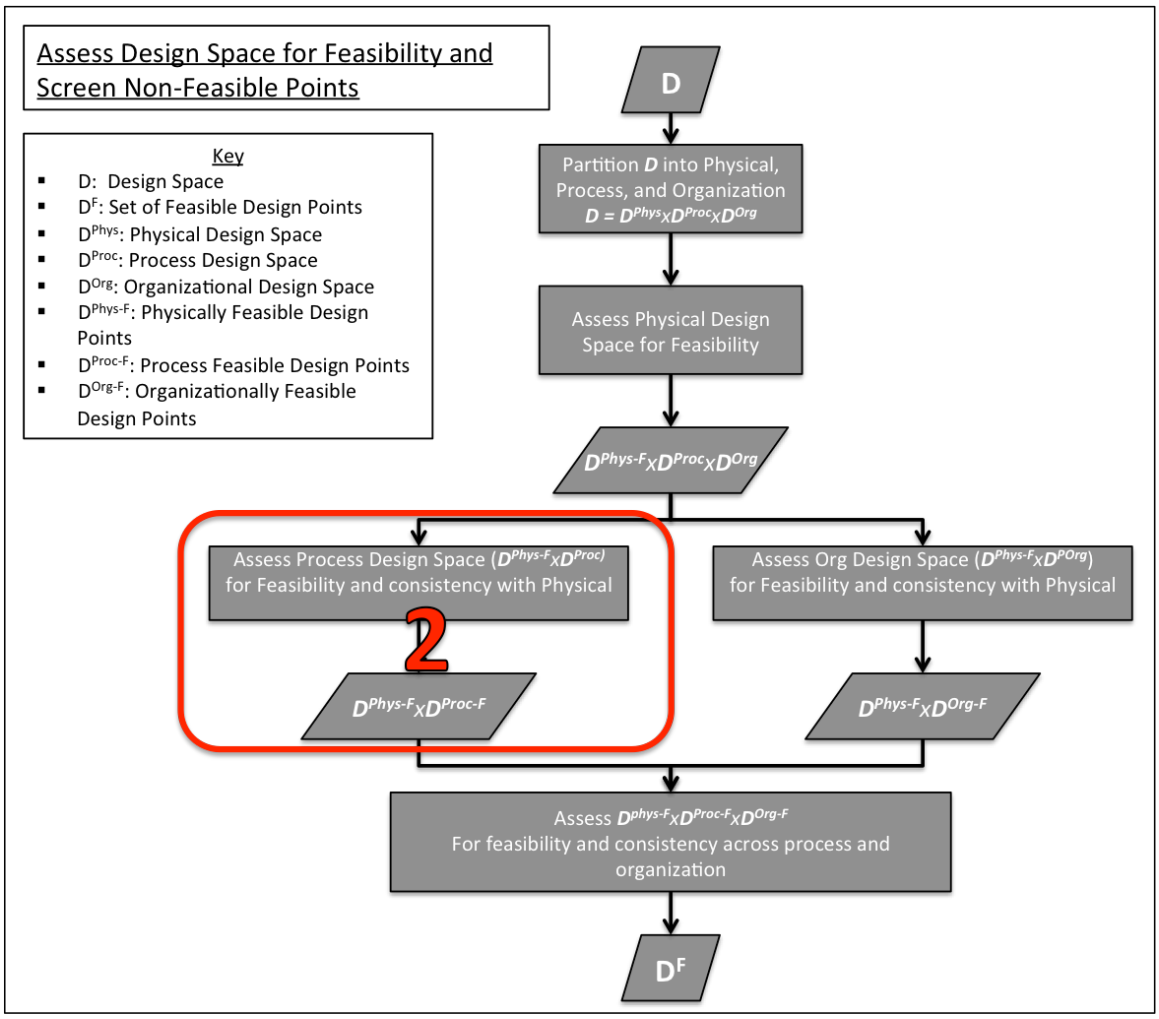


Figure 35. SoS-AFAM Step 2: Process Design Space Feasibility Analysis

The second step of the SoS-AFAM is the process design space feasibility analysis, as depicted in Figure 35. This is done by assessing design points in the space $\boldsymbol{D^{Phys-F} \times D^{Proc}}$, that is, each SoS design point defined by a physically feasible composition of an SoS crossed with each potential process architecture. The primary feasibility question to answer is: can a given set of systems conduct the required process? This makes an implicit assumption: an identified process results in the desired SoS behavior. This assumption is validated in the choice of processes that define the design space.

There are two types of process elements or factors: functional flows and rules of employment. In DODAF, these may be described by an OV-5a: Operational Activity Decomposition Tree, OV-5b: Operational Activity Model, OV-6a: Operational Rules Model, OV-6b: State Transition Description, or OV-6c: Event-Trace Description (DOD CIO 2010). Systems may use various functional flow block diagrams, IDEF0 diagrams, kill-chains, flow-charts, or other lists that describe and define rules of employment. Regardless of the precise method of describing the process architecture, a process architecture describes the necessary functions, their sequencing, and the rules of employment for a process.

a. Initial Process Feasibility Test

The initial process feasibility test simply concerns the ability of a composition of systems that form an SoS to perform the necessary functions indicated in a process architecture. To do this, one must identify the functions available to a composition of systems and the required functionality for a given process. In DODAF, the capabilities of a given system are cross-referenced with operational activities in a CV-6: Capability to Operational Activities Mapping (DOD CIO 2010). In non-DOD systems, this data may have to be recreated from another view. With this data, one may identify the operational activities each system is capable of and define, for each system, a vector that corresponds to this data (e.g., if there are three operational activities, define a binary vector of length three in which a 1 in the n^{th} position indicates that the system is capable of performing the n^{th} operational activity). This may be done similarly for each process. The algorithm simply compares the available operational activities provided by a composition of systems to the required operational activities of a given process.

```
Algorithm 2.   Initial Process Feasibility Algorithm
DEFINE the Set of Design Points
FOR Each Design Point
          DEFINE an empty SoS operational activity vector
          FOR EACH INCLUDED SYSTEM
               SUM   the   included   system's   operational
                    activity vector to the SoS operational
                    activity vector
          ENDFOR
          DEFINE an empty SoS operational activity sequence
          vector
          FOR EACH INCLUDED Operational Activity Sequence
          SUM the included operational activity sequence to
               the SoS operational activity sequence vector
          ENDFOR
          IF each entry of the operational activity vector
               (SoS available capability) is greater than
               or   equal   to   the   operational   activity
               sequence vector
               SoS Design Point is feasible, include this
                    design point in the feasible array
          ENDIF
ENDFOR
```

A simple example may clarify this algorithm. Consider a set of systems that can conduct operational activities in accordance with Table 4. This table is an extrapolation of the data for the set of potential systems that could come from a DODAF CV-6, or, in the case of non-DOD systems, other similar architecture views. Additionally, consider the set of potential operational activity sequences described in Table 5. This table is abstractly represented in Table 6. Together, these three pieces of information can be used to assess which SoS composition may complete which process.

Table 4. System versus Operational Activity

	Afghan Artillery	U.S. 155mm Artillery	Afghan TOC	American TOC	Conventional PLT	SF Team	Afghan Platoon 1	Afghan Platoon 2	UAV
Observe					X	X	X	X	X
Deconflict			X	X					
Shoot	X	X							

Table 5. Example Processes

Process 1 (P1)	Observe → Deconflict → Shoot
Process 2 (P2)	Observe (x2) → Deconflict → Shoot
Process 3 (P3)	Observe → Shoot
Process 4 (P4)	Observe (x2) → Shoot

Table 6. Minimum Functions By Process

Process	**Observe**	**Deconflict**	**Shoot**
P1	1	1	1
P2	2	1	1
P3	1	0	1
P4	2	0	1

In this case, the algorithm may consider two physically viable compositions, call them SoS-1 and SoS-2. SoS-1 includes the "Afghan Rifle Platoon – 1," "U.S. Rifle Platoon," and "SOF Team." SoS-2 includes the "U.S. Headquarters," "U.S. Artillery," and "U.S. Rifle Platoon." Both form connected networks by Table 3. and are therefore included in $\boldsymbol{D^{Phys-F}}$. SoS-1 has the capacity to conduct the "Observe" operational activity by each of its three systems by Table 4. This can be represented as a vector, $\langle 3, 0, 0 \rangle$. Compared to each process, however, this SoS composition is not feasible as it cannot "Shoot" (required for all processes) nor can it "Deconflict." Thus, all of these design points are infeasible. On the other hand, SoS-2 has the capacity to "Observe," "Deconflict," and "Shoot" once each time, represented as $\langle 1, 1, 1 \rangle$. This meets or exceeds the requirements to conduct P1 and P3, but not P2 or P4 per Table 6. Thus, the design points (SoS-2) x P1 and (SoS-2) x P3 are feasible and included in $\boldsymbol{D^{Proc-F}}$ and the others are not.

This analysis provides a high-level, low-fidelity, but quick analysis of potential SoS designs' process feasibility. The next sections examine, in greater depth, feasibility issues related to acceptance of rules of employment, and process interactions and de-conflictions.

b. Expanded Process Feasibility Test

If desired or necessary, one may develop more detailed process feasibility tests. These begin with the set $\boldsymbol{D^{Proc-F}}$ as defined in Section III.C.2.a. One can then assess for rules of employment, resource and communication flow, and process de-confliction.

If a process has a defined rule of employment, one may interview system program managers regarding their desire or ability to follow that rule. For example, in an indirect fire situation, a process rule may be that the SoS chooses targets in such a way that it maximizes the potential number of enemy killed without regard for civilian casualties. While this may be acceptable for some systems, other systems may not choose to operate with that rule in place. To do this, one must articulate the set of possible rules, which are process design points and interview the relevant system managers. This can be expressed in a table such as Table 7.

Table 7. System Acceptance of Process Rules

	U.S. Headquarters	**Afghan Headquarters**	**Continue for other systems…**
Two required observers	Acceptable	Acceptable	…
One required observer	Acceptable	Acceptable	…
Maximize enemy killed	Not Acceptable	Acceptable	…
Do not engage locations with civilian presence	Acceptable	Acceptable	…

To assess if a design point is feasible, one merely identifies the rules of employment for that design point and cross references that against the included systems for the design point and highlights any non-acceptable rules of employment making the design point infeasible. This is simple enough that it does not warrant specific pseudo-code.

The second expanded process feasibility test involves considering process conflicts. To do this, one assesses the process flow for simultaneous operational activities that must be conducted and ensuring that these simultaneous activities do not conflict.

For example, consider that when one fires an artillery round in an indirect fire scenario, one must "clear the airspace," i.e., ensure that no aircraft are operating in or near the same area as the projectile flight path.[16] Accordingly, we can consider that there is a conflict between simultaneous observation on the part of an aircraft and shooting on the part of an artillery system. This is seen in Table 8.

Table 8. Example System Process Interference

	Afghan Artillery - Shoot	U.S. Artillery - Shoot	Afghan HQ - Decide	U.S. HQ - Decide	U.S. Rifle Platoon - Observe	U.S. Special Operations Team - Observe	Afghan Rifle Platoon - 1 - Observe	Afghan Rifle Platoon - 2 - Observe	U.S. UAV - Observe
Afghan Artillery - Shoot	-	X							X
U.S. Artillery - Shoot	X	-							X
Afghan HQ - Decide			-						
U.S. HQ - Decide				-					
U.S. Rifle Platoon - Observe					-				
U.S. Special Operations Team - Observe						-			
Afghan Rifle Platoon - 1 - Observe							-		
Afghan Rifle Platoon - 2 - Observe								-	
U.S. UAV - Observe	X	X							-

In general, one may identify the set of process interferences by defining a matrix in which the rows and columns are defined by the system and each of its possible functions (e.g., "Afghan Artillery – Shoot" is one row / column, if Afghan Artillery had the ability to also observe, "Afghan Artillery – Observe" would be another row / column). Note that in this set up, a system may conflict with itself if it cannot simultaneously perform two of its own functions. After defining this process interference matrix one further defines each set of simultaneous operational activities that must occur. This is done by assessing the operational activity sequences and defining the set of functions that must be conducted in each. One then develops an algorithm to assess each design point for as follows:

[16] This is somewhat simplified. Military fire support officers and air liaison officers devote significant attention to ensuring artillery rounds do not impact aircraft. There are a variety of tactics, techniques, procedures, and information systems that are devoted to ensuring this de-confliction in the U.S. Military.

Algorithm 3. Process Deconfliction Algorithm

```
Call Process Conflict Matrix
Call Process Simultaneous Activity Sets
FOR Each Design Point
          Identify the Current Process
          FOR Each set of simultaneous activities in the
          current process
               FOR Each possible system (Identify what
               functionality is available for this SoS)
                    If that system is not included in the
                    SoS composition
                         Delete the rows and columns
                         associated with that system from
                         the Process conflict matrix
                    END If
               FOR Each pair of simultaneous activities in
               the current set, call them x and y
                    Set Conflict = 1
                    WHILE Conflict == 1
                    Search the modified conflict matrix for
                    the first non-checked system-activity
                    pair that conducts x activity
                    IF that row contains at least one
                    element in the range of systems that
                    conduct y's activity that is not a
                    conflict
                         Set Conflict = 0
                    END IF
                    IF all possible systems have been
                    checked, break
                    END WHILE
                    IF Conflict == 1
                         Identify this system as infeasible
                         Break
                    END IF
               END FOR
          END FOR
END FOR
```

Through these expanded process feasibility tests, one may identify potential conflicts or limitations of an otherwise feasible process architecture and further winnow the process architecture design space.

3. Organization Design Space Feasibility Analysis

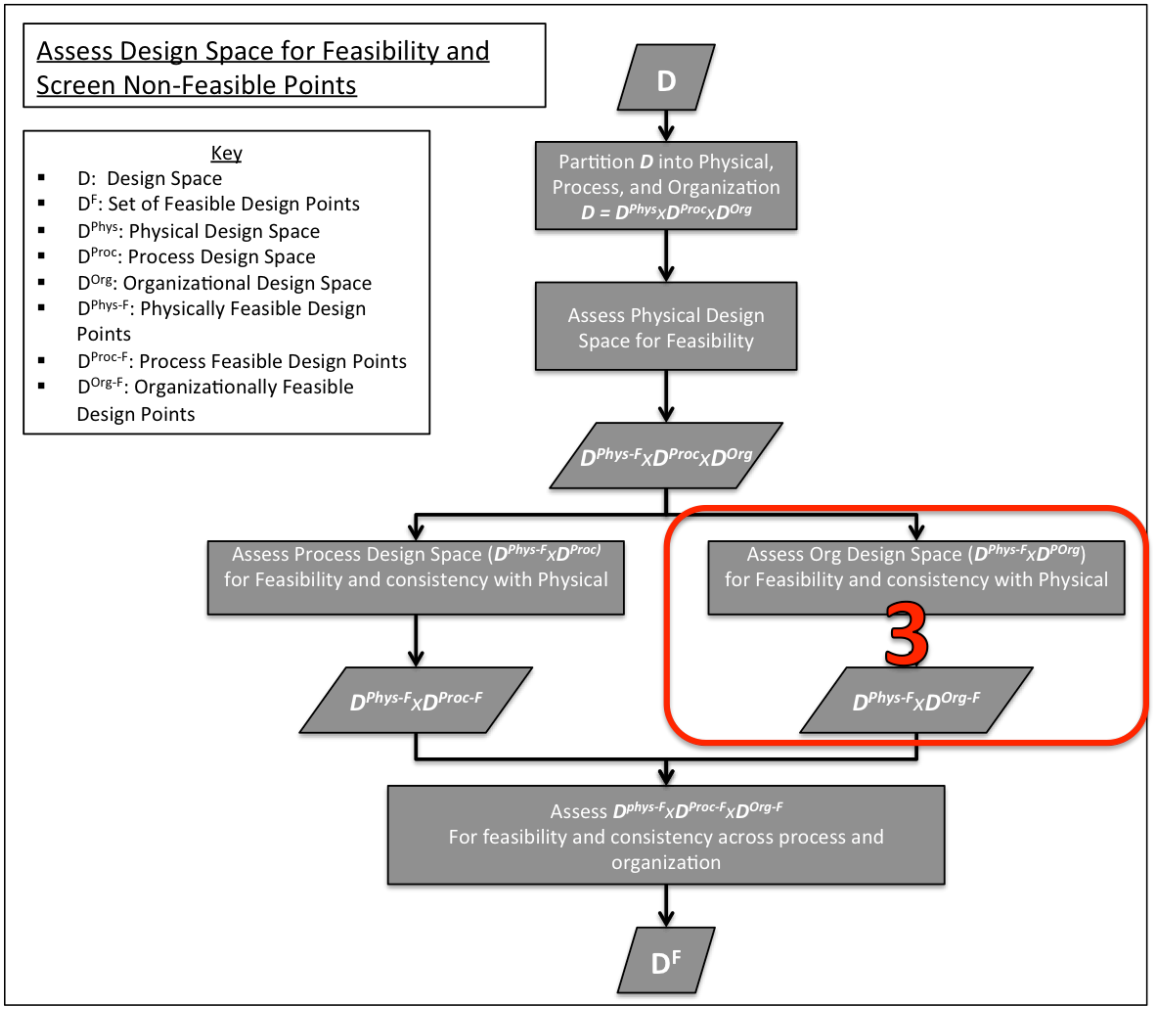


Figure 36. SoS-AFAM Step 3: Organization Design Space Feasibility Analysis

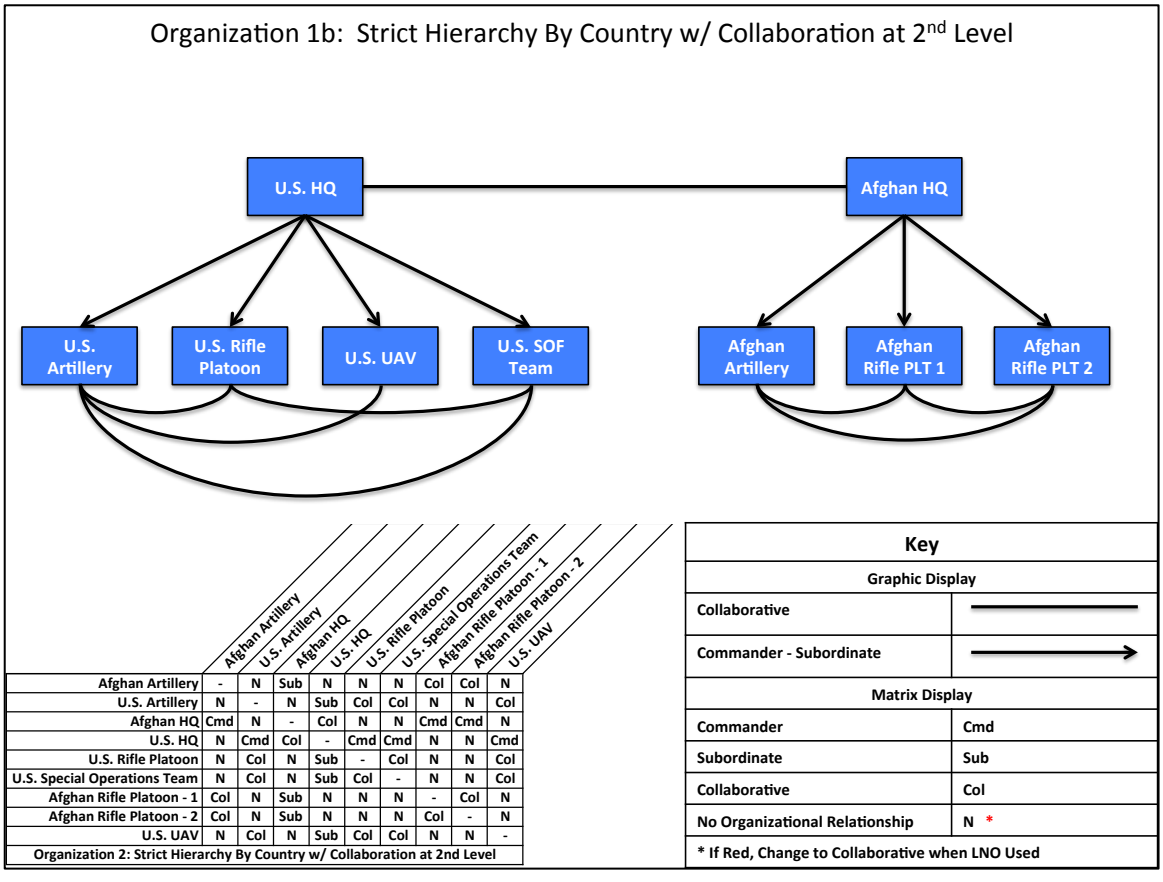
The third step of the SoS-AFAM is the Organization Design Space Feasibility Analysis as depicted in Figure 36. This test assesses design points in the design space defined by $\boldsymbol{D^{Phys-F} \times D^{Org}}$. The feasibility tests in this case answer the questions:

- Are the defined relationships acceptable to the included systems?
- Does the organization form a connected network?

- Is the organization supported by the physical architecture?

Recall that each organization, $\boldsymbol{O_i} \in \boldsymbol{D^{Org}}$ is defined as the set of relationships between each pair of potential constituent systems along with the definition of each relationship. For n potential systems, this may be expressed as a $n \times n$ matrix whose i-j entry is the relationship between the i^{th} and j^{th} systems. This is similar, although not precisely the same, as the DODAF OV-4 (DOD CIO 2010).

An example of such an organization may be seen in Figure 37. In this example, there are three possible relationships: "Commander-Subordinate" (represented by an arrow), "Collaborative" (represented by a line), or "No Relationship." This is presented both as graphic model and a matrix. This example includes every possible system that defines the physical design space.



	Afghan Artillery	U.S. Artillery	Afghan HQ	U.S. HQ	U.S. Rifle Platoon	U.S. Special Operations Team	Afghan Rifle Platoon - 1	Afghan Rifle Platoon - 2	U.S. UAV
Afghan Artillery	-	N	Sub	N	N	N	Col	Col	N
U.S. Artillery	N	-	N	Sub	Col	Col	N	N	Col
Afghan HQ	Cmd	N	-	Col	N	N	Cmd	Cmd	N
U.S. HQ	N	Cmd	Col	-	Cmd	Cmd	N	N	Cmd
U.S. Rifle Platoon	N	Col	N	Sub	-	Col	N	N	Col
U.S. Special Operations Team	N	Col	N	Sub	Col	-	N	N	Col
Afghan Rifle Platoon - 1	Col	N	Sub	N	N	N	-	Col	N
Afghan Rifle Platoon - 2	Col	N	Sub	N	N	N	Col	-	N
U.S. UAV	N	Col	N	Sub	Col	Col	N	N	-

Organization 2: Strict Hierarchy By Country w/ Collaboration at 2nd Level

Figure 37. Example Organization Definition

Furthermore, through engagement with each constituent systems' management, one can define which relationships are acceptable and which are not to that constituent system. This may be an absolute—as in a general will never consent to be commanded by a private, or conditional, as in the Special Operations team may consent to be commanded by the Afghan HQ for certain missions. The set of acceptable relationships may be defined as a matrix in a similar manner to the organization definition. With this, one can assess, for each design point in $\boldsymbol{D^{Phys-F} \times D^{Org}}$, which are acceptable to all included systems. This may be done as follows:

Algorithm 4. Initial Organization Feasibility Algorithm

```
DEFINE the set of physically feasible SoS vectors
DEFINE each organization matrix
DEFINE the nxn Acceptable Organization Matrix
FOR Each Physically Feasible SoS vector
          FOR Each Organization
               FOR i = 1 to n
                    FOR j = 1 to n
                         IF the ith and jth system are
                         included
                              IF the ij entry of the
                              current Organization is not
                              included in the ij entry of
                              the Acceptable Organization
                              Matrix
                                   DEFINE this design point
                                   as not feasible
                                   BREAK from the i and j
                                   loops
                              END IF
                         END IF
                    END FOR j = 1:n
               END FOR i = 1:n
               IF the organization was not found not
               feasible
                         Define the design point as
                         feasible
               END IF
          END FOR Each organization
END FOR Each Physically Feasible SoS Vector
```

The second question for organizational feasibility is, does the organization form a connected network? This is a feasibility requirement for reasons similar to the physical connectivity requirement—in order to work in concert to provide an emergent capability, the constituent systems in an SoS must be connected. Consider, for example, the general organization depicted in Figure 37. If that organization is applied to the set of systems that includes all of the depicted ones except for the U.S. and Afghan Headquarters, one will, in effect, see the organization depicted in Figure 38. Clearly, there are two distinct divisions to this organization making it an infeasible SoS.

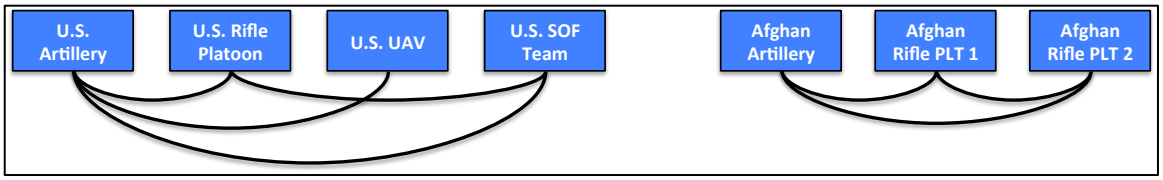


Figure 38. Example Organization with Key Systems Excluded

To assess for organizational connectivity, we consider two systems connected if they have an organizational relationship. This is done through the following algorithm:

Algorithm 5. Organizational Connectivity Algorithm

```
DEFINE the set of design points that are physically
feasible, and organizationally feasible from an
organization acceptance perspective.
FOR Each design point
          FOR i = n:1
               IF the system is not included
                    Delete the ith row and column or the
               organization matrix
               END IF
          END FOR i = n:1

          FOR each entry in the organization matrix
               IF the entry defines an organizational
               relationship
                    DEFINE that entry as a 1
               ELSE
```

```
                    DEFINE that entry as a 0
               END IF
          END FOR each entry in the organization matrix

     CALL FUNCTION "ISCONNECTED" and assess if the
          organization matrix is connected.
     IF the adjacency matrix is connected
          ADD physical SoS composition to physically
          feasible list
END FOR Each design point
```

A third test of organizational feasibility is with regard to the number and type of relationships that are acceptable for any given node. For example, if an organization has a command—subordinate relationship, one may wish to desire the maximum number of subordinates any system commands (e.g., in the Army, a common heuristic is no more than three to five subordinates) or limit the number of commanders a given system has (e.g., in the Army, the principle of unity of command would set this limit at one).[17] For example, if the organizational design is as depicted in Figure 37, then one can see that the "U.S. Headquarters" has four subordinates. This may be acceptable, but, if one wishes to limit that to three subordinates, this organizational design is not acceptable unless one of the subordinate systems is excluded from the SoS. The general algorithm for assessing this is as follows:

Algorithm 6. Organizational Relationship Limits

```
Define the maximum number of relationships for each
relationship type as R_type
FOR Each system design
          Define the organizational relationship matrix
          FOR each relationship type
               FOR Each System
                    Sum the number of times the current
                    relationship type occurs in the current
                    system's row, call this r
```

[17] Note that one may account for these principles when defining the organizational design space; however, that is not necessarily always true or desirable. One may define an organizational design in which one system is the commander of many systems, but only consider it feasible when there is a limited set of systems included in the SoS, thus limiting the number of subordinates.

```
                    If r > R_type
                         Define this system as infeasible
                         Break
                    End If
               End FOR
          End FOR
END FOR
```

After assessing for organizational acceptability and connectivity, one may assess to see if the organization is supported by the physical connectivity. That is, for every organizational relationship, there is some communication that must occur between the two systems, this communication must be supported by the physical interface between those two systems. For example, in Figure 37 the U.S. and Afghan Headquarters have a collaborative relationship; however, Table 3. indicates that these two systems only have the ability to communicate if the "Afghan Liaison" refactorization is included. Thus, the organization is only feasible if the "Afghan Liaison" is included.

The algorithm for assessing physical support of an organization involves defining the organization for a given set of systems and, for each non-zero entry (i.e., any entry that has a defined organizational relationship) in the organizational relationship matrix, assess whether 1) there is a means of physical communication and 2) whether this physical communication supports the necessary communication as defined by that relationship. This requires the connectivity matrix as defined in the physical feasibility assessment section.

Algorithm 7a. Physical Support of SoS Organization

```
FOR Each Design Point (of those thus far assessed as
physically feasible)
Call the Physical Connectivity Matrix from the Physical
Feasibility Assessment
          For i = n:1
               IF the ith system is not included in the
               design point
                    Delete the ith row and column out of
                    that design point's organization matrix
```

```
                    Delete the ith row and column from the
                    physical connectivity matrix
               END IF
          END FOR i = n:1

          FOR Each element of the organization matrix
               IF an element has a defined organizational
               relationship && does not have a physical
               connection in the corresponding Physical
               Connectivity Matrix
                    Define the design point as not feasible
               Break
          END FOR

          IF the organization has not been defined not
          feasible
               Define the design point as feasible
          END IF
END FOR Each design point
```

Much of the Algorithm 7a depends upon the definitions of an organizational relationship and physical connectivity. The highest level, low fidelity physical connectivity matrix only considers if there exists a potential physical connection of any sort. Increasing levels of fidelity vary this (as discussed in Section III.C.1) according to technical details. The type of relationship indicates both the type and amount of communication that must occur between two systems. At a detailed level, the physical connectivity must be sufficient such that it is capable of transmitting the required organizational information in a timely manner. If necessary, such detailed requirements may be articulated in the organization architecture relationship definition.

This increased level of detail may be expressed with a slight modification of Algorithm 7a by varying the statement that assess for physical support of an organizational relationship as follows:

Algorithm 7b. Physical Support of SoS Organization (Modified)

```
          FOR Each element of the organization matrix
```

```
        DEFINE organization required information
        type as O_org and bandwidth as B_org
        DEFINE the available physical connectivity
        between the relevant systems as O_phys and
        B_phys

        IF an element has a defined organizational
        relationship
             IF O_org is not equal to O_phys (or,
             O_org is not a member of O_phys if
             O_phys is a vector)
                  Define the design point as
                  infeasible
                  Break
             END IF
             IF B_org > B_phys (if the required
             bandwidth exceeds the available
             bandwidth)
                  Define the design point as
                  infeasible
                  Break
             END IF
    END FOR
```

Ultimately, the organizational feasibility assessment defines a set of points that contain physical and organizational design parameters and are feasible physically, organizationally, and in their interaction (i.e., the physical supports the organization); call the resultant space $\boldsymbol{D}^{\boldsymbol{Phys-F}} \times \boldsymbol{D}^{\boldsymbol{Org-F}}$. These design points must be assessed in conjunction with the process design points for total design point definition and assessment.

4. Total Design Space Feasibility Analysis

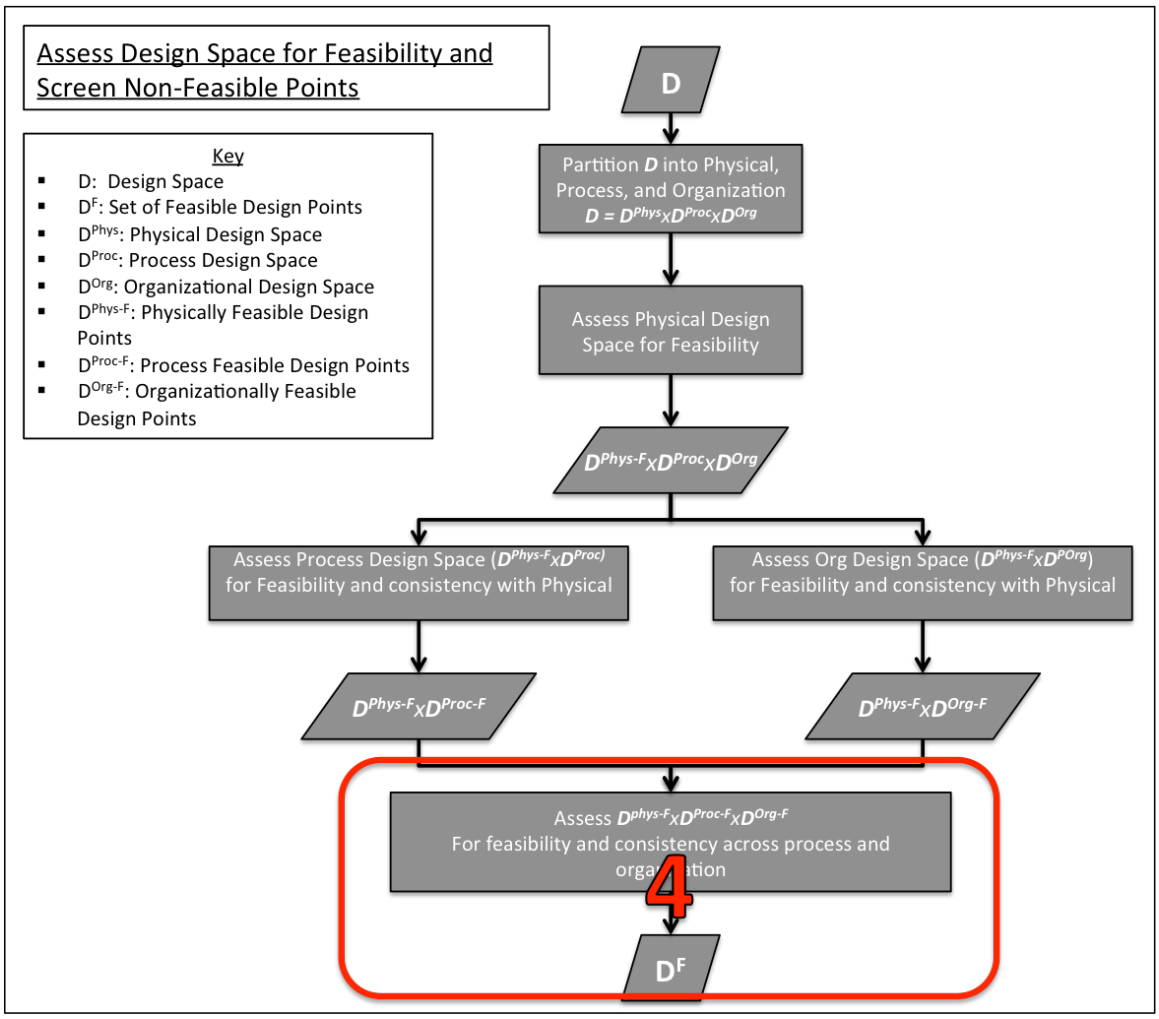


Figure 39. SoS-AFAM Step 4: Total Design Space Feasibility Analysis

The fourth and final step of the SoS-AFAM is the Total Design Space Feasibility Analysis. At this point, one has two distinct sub-design spaces, $\boldsymbol{D^{Phys-F} \times D^{Proc-F}}$ and $\boldsymbol{D^{Phys-F} \times D^{Org-F}}$. One may combine these to define the total feasible design space by considering every pair of feasible points from each of the sub design space that share a common set of physical parameters.

A minimal requirement is that the design point is feasible from all three perspectives. For example, if a design point is can take one of two physical architectures – say C1 and C2, two organizational architectures, say O1 and O2, and two process architectures, P1 and P2, there are eight possible designs. Assuming C1 and C2 are both feasible, one may consider if the designs for each organization and process combined with a physical architecture are feasible. Example results are seen in Tables 9 and 10. As one can see, the only feasible design point from all three perspectives if C1-O1-P1 as that is the only one that is feasible from both the organizational and process perspectives.

Table 9. Example Results of Process and Organization Architecture Feasibility Assessment

C1-O1	Feasible
C1-O2	Not
C2-O1	Not
C2-O2	Feasible

C1-P1	Feasible
C1-P2	Feasible
C2-P1	Not
C2-P2	Not

Table 10. Sample Combination of Process and Organization Feasibility Analysis

C1-O1-P1	Feasible
C1-O1-P2	Not
C1-O2-P1	Not
C1-O2-P2	Not
C2-O1-P1	Not
C2-O1-P2	Not
C2-O2-P1	Not
C2-O2-P2	Not

More generally, this analysis may be completed with the following algorithm:

```
Algorithm 8.   Total SoS Design Space Analysis
Call set of physical feasible designs
Call the set of process feasible designs
Call the set of organization feasible designs
FOR Each physical feasible design
          FOR Each Organization
```

```
                For Each Process
                     If the Point defined by the current
                     Organization and Physical parameters is
                     in the set of Organization Feasible
                     Designs and the Point defined by the
                     current Process and Physical parameters
                     is in the set of Process Feasible
                     Designs
                          Define the point defined by the
                          current physical, process, and
                          organization parameters as
                          feasible
                     ELSE
                          Define the point defined by the
                          current physical, process, and
                          organization parameters as not
                          feasible
                     END IF
                END FOR
           END FOR
END FOR
```

The next question to assess is if the organization supports the process. In general, if an SoS design point has been found feasible thus far, it is physically and organizationally connected and has sufficient functionality to achieve the desired process. Accordingly, as one progresses from one step to the next in an operational activity flow, information may be passed between the systems by virtue of the physical and organizational connectivity. There are two ways to further refine this question. The first is by determining a maximum distance (or time) one wishes to allow between any two operational activities. The second considers the specific information (type and amount) required between two operational activities and if one can form a path (of any length) that allows for this information per the organizational definitions.

For example, consider an SoS composed of: a "U.S. Headquarters," "Afghan Liaison," "Afghan Headquarters," "U.S. Artillery," and "Afghan Rifle Platoon – 1;" the organization depicted in Figure 37 conducting Process 3 with functionality as described

by Table 4. This SoS would be as depicted in Figure 40. [18] As the SoS is conducting Process 3, there is only one pair of functionalities to assess, "Observe" and "Shoot." We assess the distance between these two points by building a path between them. In this case, the only path is "Afghan Rifle Platoon – 1," "Afghan Headquarters," "U.S. Headquarters," then "U.S. Artillery" and is of length three. If, for some reason, a decision-maker wished to only allow for paths of length one,[19] then this would not be a feasible system as there is no shorter path than the one described.

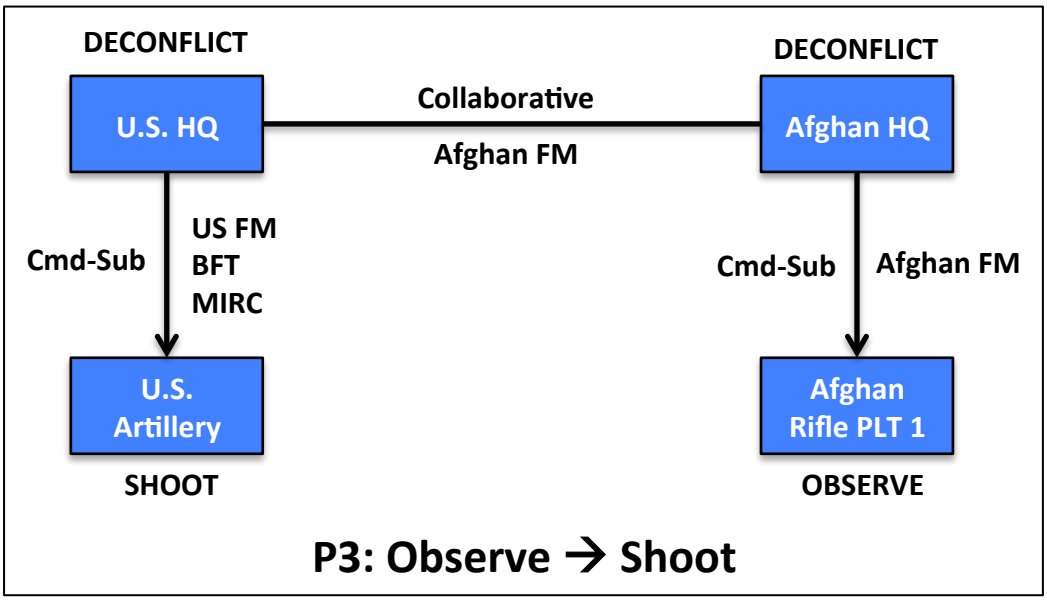


Figure 40. Example SoS For Organizational – Process Analysis

A refinement on this is to consider the time to transmit from the sender system to the receiver system. This is done by considering the size of the required message, the bandwidth at each link, and the time to re-transmit the message through a system. This requires one to identify the bandwidth of each link, the processing time of each system, and then use standard network flow algorithms (e.g., Ahuja et al., 1993). If the size of a required message exceeds the capacity of any link, the distance will be infinite, meaning

[18] Note that this is a compilation of multiple, previously introduced, views used here for convenience.

[19] A path of length one means direct, organizational communication between the two systems. This prevents the "telephone game" in which one passes information between multiple other systems before it reaches its intended target.

no path could be formed. A similar, but qualitative question is, if the organizational relationships only authorize specific information,[20] one can then do a similar analysis of the ability to send the required message along each node of a path between the sender and receiver. In general, all of these methods follow the same basic algorithm that may be modified according to the exact requirements. This is seen as follows:

Algorithm 9. Organization Support of Process Analysis

```
FOR Each design point
          Define the current physical composition as C
          Define the current process as P
          Define the current organization as O
          Define the current adjacency matrix based upon P,
          O, and C
          Define the max path length (or time, as defined
          by the decision-maker)
          FOR each pair of operational activities in P,
          call them P_start and P_end
               IF P_start and P_end have a direct
               relationship in P
                    FOR each included system that conducts
                    P_start, call this S_start
                         FOR each included system that
                         conducts P_end, call this S_end
                              Call the function[21]
                              Shortest_Path(S_start, S_end)
                              IF the result is less than
                              the max path length
                                   This point is feasible
                                   Break
                         END FOR
                    END FOR
               ELSE
```

[20] In this regard, and at a high level of fidelity, one may define specific types of communication (e.g., a Call for Fire in a specific format), and define relationships that may use or require that format.

[21] These functions are commonly available in many network science applications. The author used MATLAB networks routines published by a variety of authors and compiled by MIT at http://strategic.mit.edu/downloads.php?page=matlab_networks

```
                    (No need to check, progress to the next
                    pair)
               END IF
          END FOR
          IF All pairs of P_start and P_end have a path
          that is less than the max length
               Define the system as feasible
          ELSE
               Define the system as infeasible
          END IF
END FOR
```

As with the other analyses, it makes the most sense to progress from the highest level, lowest fidelity tests to lower level, high fidelity tests in sequence. This is because the high level tests are generally quicker and have the potential to remove a significant number of infeasible design points that do not warrant greater fidelity testing.

After any level of fidelity testing has been completed, the end result is a set of design points that describe feasible SoS from the perspectives of physical, process, organization, and their interactions.

5. SoS-AFAM Conclusion

The end result of the SoS-TDM Design Space Definition and Design Space Analysis, the SoS-AFAM, is a significantly reduced subset of the initial SoS design space defined by the fact that each design point in it is feasible, from a physical, process, and organization perspective. Call this space:

$$\boldsymbol{D^F} = < \boldsymbol{d} \in \boldsymbol{D} | \boldsymbol{d}\ is\ feasible >$$

Each design point is a set of parameters for an SoS that, together with the constituent system data, inform a unique SoS architecture that may be built and includes SoS physical, process, and organizational perspectives. Moreover, these design points serve as inputs to subsequent system analysis models (e.g., operational models or cost models) conducted in the fourth step of the SoS-TDM.

D. SOS-TDM – FEASIBLE DESIGN SPACE ANALYSIS

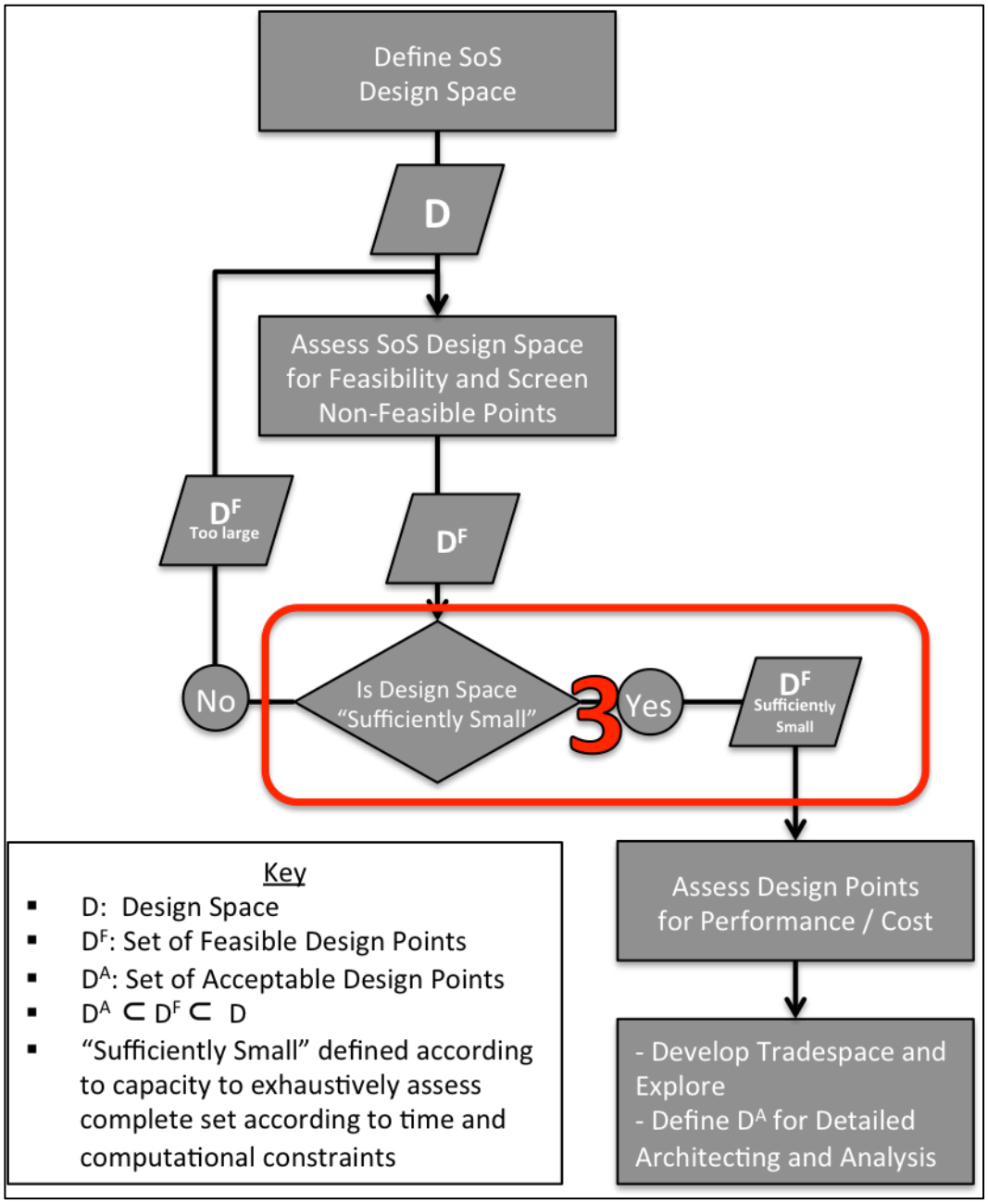


Figure 41. SOS-TDM – Feasible Design Space Analysis

The third step of the SoS-TDM is Feasible Design Space Analysis, highlighted in red and seen in Figure 41. The input to this step is $\boldsymbol{D^F}$. It is a trivial matter to assess its size. It is a discrete set of parameter vectors that are held in a matrix or similar data store in one's computer code; an engineer merely looks up the appropriate size metric.

Defining "sufficiently small" is only slightly more complicated. This depends upon the time available and the time to compute an appropriate number of runs of an operational simulation (or multiple distinct simulations) for a design point. Sufficiently small is, therefore a number *s*, such that

$$s \leq \frac{Time\ Available}{Assessment\ Time\ Per\ Design\ Point}$$

If $|\boldsymbol{D^F}| \leq s$, then it is sufficiently small and may be assessed exhaustively in the next step of the SoS-TDM. If $|\boldsymbol{D^F}| > s$, one must iterate the SoS-TDM Design Space Analysis step at a higher level of fidelity and with stricter requirements for feasibility. This will produce a new $\widetilde{\boldsymbol{D}}^{\boldsymbol{F}} \subset \boldsymbol{D^F}$. One then assesses $\widetilde{\boldsymbol{D}}^{\boldsymbol{F}}$ in the same manner as described in Section III.C. Iterate these steps as necessary until a sufficiently small feasible design space is defined.

E. SOS-TDM – DESIGN POINT ASSESSMENT AND TRADESPACE ANALYSIS

The final step of the SoS-TDM is to exhaustively assess the set of feasible design points for performance (according to pre-defined, desired MOEs) and use this for tradespace exploration. For context within the general SoS-TDM, this step is highlighted in red in Figure 42. This step of the SoS-TDM has two primary components, design point assessment and tradespace development and exploration. The end result is a tradespace that may be explored and used to define the set of acceptable design points, $\boldsymbol{D^A}$. These are the design points that are both feasible and satisfy decision-makers' requirements. Within $\boldsymbol{D^A}$ there is a subset that are Pareto optimal that may be selected for subsequent detailed architecting and analysis.

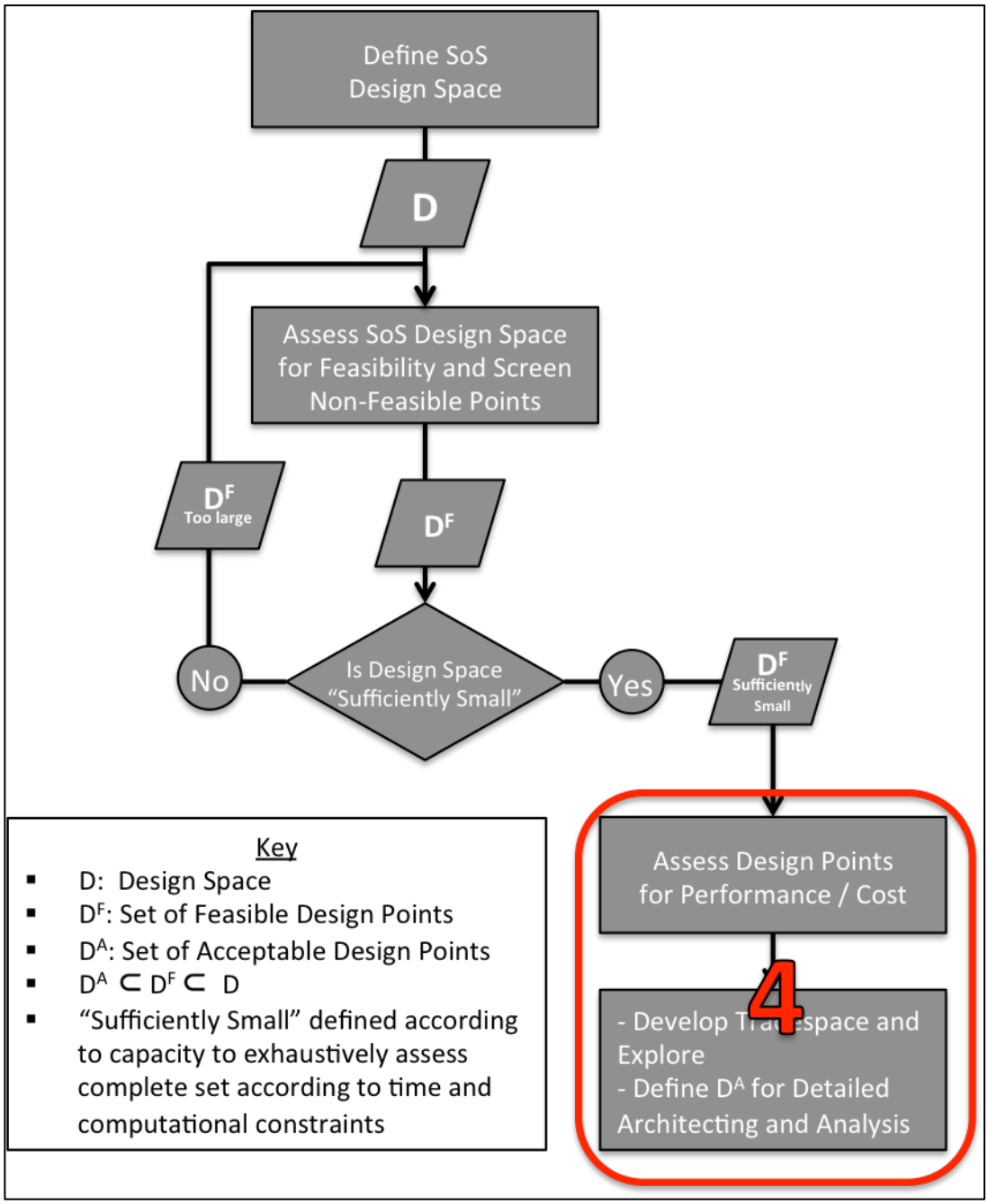


Figure 42. SOS-TDM – Design Point Assessment and Tradespace Analysis

The first step is to assess the design points for performance. This may be done using any number of appropriate modeling and simulation techniques, although, for SoS, the most common are ABM. Each design point is input into the chosen model or simulation and executed. If the simulation is non-deterministic, it is repeated an appropriate number of times to achieve statistically accurate results (30 repetitions is standard). Significantly, the design points in $\boldsymbol{D^F}$ that are the inputs to the simulations have varying physical, process, and organization parameters that affect the results of a simulation and provide more realistic results. Methods of modeling and simulation of SoS are well developed for individual design points; see Section II.E.2 for a more detailed discussion of SoS analysis. Once the simulations are complete, one may develop a tradespace.

The second step is the actual presentation of a tradespace. The specific presentation is contingent upon the desires of a decision-maker; however, it should demonstrate the relationship between a set of design parameters for a system and the resulting output. Decision-makers can then vary the desired requirements that define the set of acceptable points, either in terms of system attributes (e.g., system performance, such as cost) or system parameters (i.e., the inclusion or exclusion of a system) as discussed in Chapter II. This defines $\boldsymbol{D^A}$.

Once a decision-maker defines $\boldsymbol{D^A}$ an engineer may then assess the set of Pareto optimal points within $\boldsymbol{D^A}$, and conduct detailed SoS architecting and analysis for final SoS design decision-making. At this point, the SoS-TDM is complete and one continues with the chosen SoSE and MBSE engineering processes.

F. SOS-AFAM ANALYSIS

The SoS-AFAM defines an SoS design space with parameters that describe the physical, process, and organizational structure of a potential SoS. Inherently, this increases the size of the design space. The significant question is, how quickly can we analyze this space and use it to develop a tradespace? There are two considerations by which to assess the SoS-AFAM, the number of design points that must be assessed and how quickly each design point may be assessed.

1. Number of Design Points to Assess

Recall the design space is defined as: $\boldsymbol{D} = \boldsymbol{D^{Phys}} \times \boldsymbol{D^{Proc}} \times \boldsymbol{D^{Org}}$. This has a magnitude $|\boldsymbol{D}| = \left|\boldsymbol{D^{Phys}}\right| \cdot |\boldsymbol{D^{Proc}}| \cdot |\boldsymbol{D^{Org}}|$. Recall that the magnitude of each of these subspaces is defined by the inputs:

- $2^n \leq \left|\boldsymbol{D^{Phys}}\right| \leq M_s^n$ where n is the number of distinct systems, and M_s is the maximum number of re-factorizations a system may take plus the options of no system included or the system as is
- $2^{a+b} \leq |\boldsymbol{D^{Proc}}| \leq M_f^a \cdot M_r^b$, where a is the number of non-mutually exclusive operational activity flows, b is the number of non-mutually exclusive rules of employment, and M_f *and* M_r are the maximum of any given set of mutually exclusive operational activity flows or rules of employment, respectively
- $\left|\boldsymbol{D^{Org}}\right| = O$ for some pre-defined set of heuristically chosen organizations. Recall, the alternative of considering every combination of possible relationships for r defined relationships and n systems was $r^{n(n-1)}$

Accordingly, just considering the lower bounds of the first two and only one organization, the number of possible SoS designs, $|\boldsymbol{D}|$, exceeds:

- 100,000 no later than when $n+a+b = 17$
- 1,000,000 no later than when $n+a+b = 20$
- 10,000,000 no later than when $n+a+b = 24$
- 100,000,000 no later than when $n+a+b = 27$
- 1,000,000,000 no later than when $n+a+b = 30$

Therefore, it is not tenable to consider every single design point, particularly for a complete operational simulation. The SoS-AFAM addresses this problem by partitioning the design space into three sub-spaces: the physical, process, and organizational, and then aggregating the results. This significantly reduces the number of design points that must be checked.

For convenience, let $C = \left|\boldsymbol{D^{Phys}}\right|, P = |\boldsymbol{D^{Proc}}|, and\ O = |\boldsymbol{D^{Org}}|$. Also, assume each is greater than or equal to two as the purpose of the SoS-TDM is to assess varying

parameters of each type. Initially, the number of design points one must consider is, therefore, *COP*.

The first step of the SoS-AFAM is to assess the physical design space. One must assess *C* design points. This results in some percentage of design points being feasible, say *x* percent.

The second step of the SoS-AFAM is to assess the process design space. The size of this design space is $\left|\boldsymbol{D^{Phys-F}}\right| \cdot P = xCP$ by the previous analysis. This results in some percentage, *y,* of design points that are feasible, or *yxCP* design points.

The third step of the SoS-AFAM is to assess the organizational design space. The size of this design space is $\left|\boldsymbol{D^{Phys-F}}\right| \cdot O = xCO$ by the previous analysis. This results in some percentage, *z,* of design points that are feasible, or *zxCO* design points.

The fourth step of the SoS-AFAM is to assess the total design space. One must only assess those points that have the potential to be feasible. The set of points to check are either of the form (process feasible design point)x(all possible organizations) or (organization feasible design point)x(all possible organizations) as in order to be totally feasible, each design point must be both organizationally and process feasible (these points include physical feasibility already). Thus, the number of design points that we must check is the *min((zxCO)P, (yxCP)O)*. In general, call this number, *wxCOP,* where *w=min(y, z)*.

Thus, to assess the entire design space for feasibility, we must only assess *C+xCO+xCP+wxCOP*. Let Π denote the percentage of design points in the design space one must assess. This is:

$$\Pi = \frac{C + xCO + xCP + wxCOP}{COP} = \frac{1}{OP} + \frac{x}{P} + \frac{x}{O} + wx$$

Recall that *P* and *O* are both greater than or equal to two, thus:

$$\Pi \leq \frac{1}{4} + \frac{x}{2} + \frac{x}{2} + wx = 0.25 + x + wx$$

While this may seem initially somewhat large, note that as the size of the design space increases as a function of additional processes and organizations, the percentage of design points one must assess decreases significantly. For example, if *O=P=10,* $\Pi = \frac{1}{100} + \frac{x}{10} + \frac{x}{10} + wx = 0.01 + \frac{x}{5} + wx$. Thus, as the design space increases as a function of the number of organizations and processes, the percentage of the design space we must assess is primarily dominated by the percentage of physically feasible SoS design points.

2. Algorithm Analysis

a. Physical Design Points

Algorithm 1 assesses the physical design space for connectivity. Each input into this test is a set of at most *n* systems that form an adjacency matrix. To define this matrix requires a set number of steps for each pair of possible systems, thus this first step is $O(n^2)$. Reingold (2008) showed that connectivity can be assessed as *O(log(n))* for a network with *n* nodes. Together, each design point may then be assessed in $O(n^2 + log(n))$. The detailed physical tests differ from the initial test with regard to the definition of connectivity, but, algorithmically, they are effectively the same.

One must assess $2^n \leq C \leq M_S^n$ points in this manner. For limited numbers of possible refactorizations, this number is approximately 2^n, thus the entire physical design space may be assessed in $O\left(2^n(n^2 + \log(n))\right)$.

b. Process Design Points

Algorithm 2 considers the number of operational activities available to the set of systems. If there are *f* operational activities one must simply add these numbers for each of the *n* systems, therefore this is *O(nf)*. This must be checked for each of the *xCP* process design points. *xC* is a fixed number; *P* is bounded by 2^a and M_f^a for *a* is the number of non-mutually exclusive operational activity flows (e.g., if one is choosing one operational activity flow among a set of several, then *a* would be one. If one chooses one operational activity flow from one set and another from another set, *a* would be two). As

we typically only choose one potential operational activity flow for the entire ABM, this number is most typically, M_f. Thus, this is assessed in approximately $O(xCM_fnf)$.

The second process check assesses which collections of systems accept the chosen set of rules. There was no need for a formal algorithm due to the simplicity; however, if there are at most $bM_e = r$ rules of employment and n systems, at most, we must check each system against each rule of employment, thus this algorithm is assessed in $O(nr)$. Again, there are xCP design points that must be assessed each time, thus, to assess each design point, the analysis is approximately $O(xCPnr)$.

Algorithm 3 assesses process conflicts for a set of systems and a given operational activity flow. If there are c conflict points (meaning we have identified c pairs of operational activities that must be conducted simultaneously), we must assess, at most, each pair of systems in the SoS against these conflict points. This may be done in $O(cn^2)$. We do this for each of the xCP design points, thus the entire assessment may be done in approximately $O(xCPcn^2)$.

c. Organization Design Points

Algorithm 4 assesses each organization design point for relationship acceptability. At most, this requires assessing each possible pair of systems in the SoS against the relationship acceptability matrix. This may be done in $O(n^2)$. We check this against the xCO organizational design points. Thus, the entire organization design space may be checked in $O(xCOn^2)$.

Algorithm 5 assesses organizational design points for connectivity. As the organizational matrix is already defined, we must only modify it to delete the rows and columns associated with non-included systems. This involves a set number of steps for each of the n systems, and thus may be done in $O(n)$. We then assess the resultant adjacency matrix for connectivity, which may be done in $O(log(n))$ (Reingold 2008). Thus, each point may be assessed in $O(n+log(n))$. We must assess xCO points, thus the entire analysis may be done in $O(xCO(n+log(n)))$.

Algorithm 6 assesses the number of each type of relationship each system has and compares that number against a pre-defined threshold. This therefore requires assessing each of n systems against R relationship types (e.g., collaborative, command). This may be done in $O(nR)$. We assess this for each of the xCO design points. The total analysis is therefore, $O(xCOnR)$.

The final organizational feasibility test, Algorithm 7a and 7b, assesses the physical support of each organizational relationship. Algorithm 7a assesses only against the requirement to have a communication system of any type supporting an organizational relationship; Algorithm 7b has a stricter requirement—that specific communications capabilities support organizational relationship requirements in a pre-defined manner. In either event, the assessment requires a fixed number of steps to assess each of the n^2 potential system-system relationships. Each design point may be assessed in $O(n^2)$. Again, there are xCO design points; the total analysis is, therefore, $O(xCOn^2)$.

d. Total Design Space

Algorithm 8 defines each design point that is feasible from a physical, process, and organizational perspective. Its inputs are the set of feasible process designs (of the form (physical parameters)x(process parameters)) and set of feasible organization designs (of the form (physical parameters)x(process parameters)). One must check either the organization design against each possible process or vice versa resulting in design points of the form (physical parameters)x(organization parameters)x(process parameters). One checks each of these points to ensure the (physical parameters)x(process parameters) and (physical parameters)x(organization parameters) are both feasible. This involves a fixed number of steps for each of the design points and thus may be done in $O(wxCOP)$, where w is as defined as in Section III.F.1.

Finally, if desired, one may assess the design space as described in Algorithm 9 to see if the organization supports the processes. This involves at most, assessing, the shortest path between any two systems conducting any two pairs of operational activities in the SoS. There are n systems and f operational activities, thus the set of pairs may be assessed in $O(n^2f^2)$. For each of these, one must assess the shortest path (along the

organizational adjacency matrix) that may be assessed using the Dijkstra algorithm, or a variation of it. In general, one may solve a shortest path algorithm in (Ahuja et al. 1993, 123):

$$O\left(\min\left\{m + n \cdot \log(n), m \cdot \log(\log(L)), m + n\sqrt{\log(L)}\right\}\right)$$ [22]

where n is the number of nodes in the network, m is the number of links, and L is the maximum arc length (or weight) in the network. The reason for the varying run-time measures is due to different implementations of Dijkstra's algorithm (Ahuja et al. 1993). Thus, for each design point, the organizational support of the process may be assessed in:

$$O\left(\mathrm{n}^2\mathrm{f}^2 \cdot \min\left\{m + n \cdot \log(n), m \cdot \log(\log(L)), m + n\sqrt{\log(L)}\right\}\right)$$

This must be assessed for each point that is physically, organizationally, and process feasible, which is the result of Algorithm 8. This is some percentage of the *wxCOP* design points; call it *t.* The entire design space may be assessed therefore in:

$$O\left(twxCOPn^2f^2 \cdot min\left\{m + n \cdot log(n), m \cdot log(log(L)), m + n\sqrt{log(L)}\right\}\right)$$

3. False Positives

The SoS-AFAM is vulnerable to false-positives. That is, a design point may be assessed as feasible, when in reality it is not. This is not a statistical error, rather an inherent limitation of the SoS-AFAM. Passing each of the SoS-AFAM's tests is necessary, although possibly not sufficient, for a design point to be feasible. Unfortunately, there is no way to comprehensively define a sufficient set of feasibility criteria for all systems. The SoS-AFAM is, however, resilient to false-negatives. That is, if one accepts the various tests as necessary for feasibility, the SoS-AFAM will not identify a system as infeasible when it is, in reality, feasible.

For tradespace development, it is preferable to have false-positives over false-negatives as one wishes to explore the largest set of possible system designs. False-negatives, excluded from a tradespace, have no potential to be chosen a useful design.

[22] Note: Ahuja et al. (1993, 123) use a *C* versus the *L* written here. The author modified this to avoid confusion with the *C* indicating the number of physical SoS compositions that may be formed.

False-positives, however, if chosen, will be identified as infeasible during more detailed architecting and analysis.

4. Non- Physical, Process, or Organization Interactions

It is not generally possible to categorically say that every type of interaction within an SoS may be categorized as physical, process, or organization. If an SoS has a significant interaction that falls outside of one of these categories, it would be missed. This affects both the feasibility analysis, i.e., one could generate a false positive, and the subsequent performance analysis, i.e., one would not correctly represent the SoS by missing a potential interaction, which would affect the results of a simulation. The former case is addressed in the earlier false-positive section. The latter case is generally problematic for SoSE. Identifying every possible interaction *a priori* is difficult, if not impossible. For this reason, practitioners have methods such as the "wave model" (Dahmann et al. 2011) to iterate the SoS engineering process.

5. SoS-AFAM Analysis Conclusion

The SoS-AFAM analyzes the design space of an SoS problem using a series of algorithms that assess the feasibility of a given SoS design. Significantly, by partitioning the SoS design space into four distinct spaces, the percent of design points that one must assess for feasibility is:

$$\Pi = \frac{1}{OP} + \frac{x}{P} + \frac{x}{O} + wx$$

where O is the number organizational points, P is the number of process points, x is the percentage of physical compositions that are feasible, and w is the minimum of the percentage of organizational or process designs that are feasible. There is no general method to prove what x or w are; however, it is clear that the SoS-AFAM significantly reduces the size of the design space if one can significantly reduce at least one of the three sub-spaces. Furthermore, the analysis of each design point may be assessed for its computational complexity as described in Section III.F.2 and tabulated in Table 11.

Table 11. SoS-AFAM Algorithm Analysis

Test	Analysis of a Design Point	Maximum Number of Design Points to Assess
Physical Connectedness (Algorithm 1)	$O(n^2+log(n))$	$2^n \leq C \leq M_s^n$
Sufficient Functionality (Algorithm 2)	$O(nf)$	$xCP, 2^{a+b} \leq P \leq M_f^a M_e^b$
Rule Acceptance	$O(nbM_e)$	$xCP, 2^{a+b} \leq P \leq M_f^a M_e^b$
Operational Activity Deconfliction (Algorithm 3)	$O(cn^2)$	$xCP, 2^{a+b} \leq P \leq M_f^a M_e^b$
Organization Acceptance (Algorithm 4)	$O(n^2)$	xCO
Organization Connectedness (Algorithm 5)	$O(n+log(n))$	xCO
Maximum Relationship Capacity (Algorithm 6)	$O(nR)$	xCO
Organization Supported by Physical Communication (Algorithms 7a and 7b)	$O(n^2)$	xCO
Mutual Organization and Process Feasibility (Algorithm 8)	$O(1)$	$wxCOP$
Maximum Organizational Path Length Between Any Two Operational Activities (Algorithm 9)	$O\left(\mathrm{n}^2\mathrm{f}^2 \cdot \min\left\{m + n \cdot \log(n), m \cdot \log(\log(L)), m + n\sqrt{\log(L)}\right\}\right)$	$wxCOP$

a = number of sets of mutually exclusive operational activity flows
b = number of sets of mutually exclusive rules of employment
f = number of operational activities
c = number of operational activity conflicts
C = number of physical system designs
L = maximum arc length in a network
m = number of edges (links or arcs) in a network
M_e=most number of mutually exclusive rules of employment
M_f=most number of mutually exclusive operational activity flows
M_s=most number of system refactorizations
n = Number of systems
O = number of organizational system designs
P = number of process physical system designs
R = number of relationship types
w = minimum of percent organizationally feasible or process feasible systems
x = percent of physically feasible systems

G. CONCLUSION

The SoS-TDM is a methodology to define a comprehensive SoS design space, assess it for feasibility, and use this assessment to define a "sufficiently small" subset of the design space for exhaustive performance analysis. This allows engineers to define an SoS tradespace and explore this tradespace with decision-makers. Through TSE, engineers and decision-makers may define a small subset of the feasible design space that is acceptable. Within that acceptable design space, engineers may identify and analyze the Pareto optimal design points and conduct subsequent detailed SoS architecting and analysis. This process is described in Figure 43. Ultimately, the SoS-TDM process facilitates the selection of a high level SoS architecture (that may be described in a number of manners, one notable SoS architecture framework being DODAF). By exploring a large design space that includes process and organizational considerations in addition to physical considerations, engineers are able to 1) better assess design points for feasibility across a wide range of considerations, 2) better represent the system performance attributes (e.g., operational performance, cost), and 3) better inform decision-makers of the SoS tradespace.

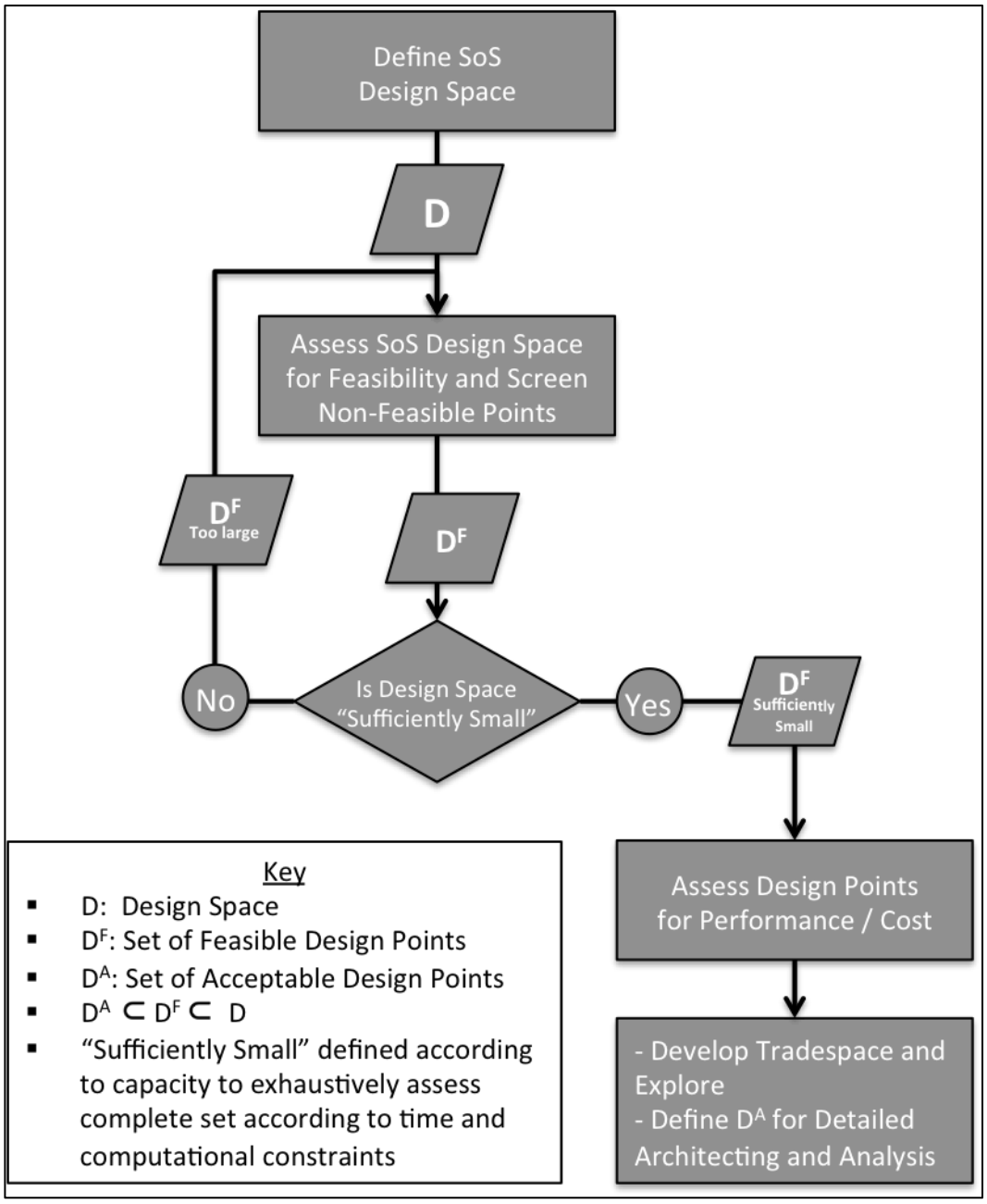


Figure 43. SOS-TDM Process

SoS architecting and analysis is challenged, from a tradespace development perspective, by large design spaces that are, in general, characterized by significant dependencies among the parameters that indicate the complex interactions among the systems. This makes it a challenge to approximate how various configurations will perform using various statistical tools. Moreover, one must still assess each potential

design for feasibility. An efficient feasibility test allows one to broadly assess the design space for feasibility and winnow infeasible points. Subsequently, one may exhaustively assess the subset of the design space that has the potential to be realized. This is done in the SoS-TDM in the step "SoS-TDM – Design Space Analysis" in which the feasibility of a design point is assessed according to its physical, process, and organizational parameters through the SoS-AFAM.

The SoS-AFAM provides a series of general tests that may be used to assess a generic SoS design for feasibility. The majority of these tests only assess one aspect of the SoS, and therefore, this significantly reduces the number of design points one must assess. Furthermore, each test may reduce the number of design points one must assess in the subsequent test (e.g., if a design point is not both organizationally and process feasible (Algorithm 8), it does not need to be assessed for maximum organizational path length between two operational activities (Algorithm 9)). In total, this allows one to quickly assess the entire design space for feasibility as articulated in Section III.F and Table 11.

This chapter presents two contributions to the state-of-the-art of MBSE and SoSE. Within MBSE, the general methodology, as seen in Figure 43, is an expansion on general MBSE design space and tradespace methodologies. Specifically, it expands these methods by including process and organization parameters to the design space. Within SoSE, it provides a non-heuristic, non-normative methodology to define a large SoS design space that includes physical, process, and organization architecture parameters and use that design space for tradespace analysis. The specific methodology for assessing for feasibility, the SoS-AFAM, is, itself, a contribution, that provides a means for iteratively assessing SoS designs for feasibility. Chapter IV presents an example implementation of the SoS-TDM.

IV. PRACTICAL IMPLEMENTATION OF THE SOS-TDM—AN EXAMPLE OF INDIRECT FIRE

This chapter demonstrates an implementation of the SoS-TDM in the development of a combined, joint[23] indirect fire (IDF) SoS. The purpose is to demonstrate the SoS-TDM and SoS-AFAM through a concrete example.

The problem is how to design an IDF SoS that maximizes enemy destruction, minimizes collateral damage, and minimizes cost. The available systems are from the U.S. Army, U.S. Special Operations, U.S. Air Force, and Afghan Army. The example itself is notional, meaning that no specific stakeholder has requested this analysis, but it is representative of many situations faced by the DOD, particularly in the context of joint, interagency, and combined operations. This represents an acknowledged SoS as the systems recognize the need to conduct the IDF mission, but the various services and commands maintain a level of control over their systems.

Through the use of the SoS-TDM and SoS-AFAM, we are able to define the IDF SoS tradespace for exploratory analysis. The design space is formed of nine possible systems with one possible refactorization, eight possible processes, and 11 possible organizations. This results in 90,112 design points. Through the use of the SoS-AFAM, 7,980, or about 9%, are found to be feasible. We then assess those 7,980 points for their performance measures, and use these results to define the SoS tradespace. Each design point can be used to define an SoS architecture that includes physical, process, and organization perspectives. Through this, engineers and analysts may define the set of acceptable design points for refined architecting and analysis.

[23] The DoD uses the terms “joint,” “interagency,” and “combined” for specific purposes. Joint indicates two or more military services (e.g., Navy and Army). Interagency indicates two or more agencies of the U.S. Government (e.g., DoD and Department of State). Combined indicates two or more allies (e.g., the U.S. and U.K.). (Joint Chiefs of Staff. 2010. “Department of Defense Dictionary of Military and Associated Terms. (JP 1-02)” Washington, DC: Department of Defense.

A. IDF SOS-TDM PROBLEM DEFINITION AND SCOPE

To employ the SoS-TDM, one must have the requisite inputs and meet the necessary assumptions defined in Section III.A. We assume that the IDF SoS is an acknowledged SoS that has only information interfaces and the performance of the systems is well understood as outlined in Section III.A. The inputs, "Valid SoS Need and Associated MOEs" and "Potential Systems, Processes, and Organizations" are defined in the next two sections.

1. Valid SoS Need and Associated MOEs

a. SoS Need and Problem Definition

We can assume that the need for an IDF SoS is valid. IDF is defined as "Fire delivered at a target which cannot be seen by the aimer" (North Atlantic Treaty Organization [NATO] 2015, 2-I-3). To do this, one must have an observer view the target and send that information to a shooter.[24] Between the observer and shooter, one may analyze the information to validate the target and other safety considerations. An IDF SoS is one that integrates the capabilities of observers and shooters via communication and information processing to provide aimed indirect fire on enemy targets.

b. Performance Measures

For this problem, there are two MOEs, one concerning how well the SoS destroys enemy targets and another concerning how well it limits collateral damage, and one MOP, its cost. Furthermore, there is the implicit MOP that the SoS is feasible; this is true of every system assessed as the SoS-TDM uses this requirement to choose systems for assessment.

The first MOE is the percent of targets destroyed (PTD). In every operational simulation, there are a known number of targets that present themselves and a number that are engaged and destroyed by the SoS. The PTD is the ratio of these two numbers:

[24] Fires—fire support and fire delivery—is certainly more complex in its details than the simple definition presented here. For deeper discussion, see U.S. Army Field Manual 3-09, "Field Artillery Operations and Fire Support" (2014) or the joint equivalent, Joint Publication 3-09, "Joint Fire Support" (2014).

$$PTD = \frac{Total\ Number\ of\ Enemy\ Targets\ Destroyed}{Total\ Number\ of\ Enemy\ Targets\ Possible}$$

The PTD has a range of zero to one, with the goal to maximize it. This MOE is chosen over other possible variations as it encompasses the entirety of the ability of the SoS. For example, percent of targets destroyed out of targets engaged gives a false understanding of the SoS's ability to destroy all targets it sees and gives the SoS an incentive to not engage difficult to destroy targets. Another alternative MOE would be percent of targets destroyed of targets that are observed; again this gives a false incentive to an SoS to not observe targets. Finally, note that there is no distinction or weighting for any higher or lower priority targets, all enemy targets are equally important.

The second MOE is the percent collateral damage (PCD). This MOE measures the percent of potential neutral targets that are damaged by the SoS. PCD is measured as:

$$PCD = \frac{Total\ Number\ of\ Non-Enemy\ Targets\ Destroyed}{Total\ Number\ of\ Non-Enemy\ Targets}$$

The PCD has a range of zero to one, with the goal to minimize it. The PCD is chosen over alternative collateral damage MOEs for reasons similar to those described for the PTD, so as to not create false incentives to not identify civilian targets or casualties.

The final measure is the cost of the system. This is measured in dollars, with the goal to minimize it. It is assessed according to a model that aggregates cost based upon the cost of each individual system, the cost of the interfaces required for that system and organization, and the cost to train the system to perform the required processes.

These three measures provide the SoS decision-maker with a tradespace. An ideal SoS design point would have a PTD = 1.0, a PCD = 0.0, and a cost of $0.00. Of course, this point is unlikely to exist and, among the many potential design points, there will be a tradeoff among the various measures.

2. Potential Systems, Processes, and Organizations

a. Systems

For the IDF-SoS there are nine potential constituent systems from four distinct commands; one of these systems may be refactored for an additional communications capability. The nine potential systems are:

- System 1 – Afghan Army Artillery Battery
- System 2 – U.S. Army Artillery Battery
- System 3 – Afghan Army Kandak (Battalion) Headquarters
- System 4 – U.S. Army Battalion Headquarters
- System 5 – U.S. Army Rifle Platoon
- System 6 – U.S. Special Operations Forces Team
- System 7 – Afghan Army Rifle Platoon 1
- System 8 – Afghan Army Rifle Platoon 2
- System 9 – U.S. Air Force Unmanned Aerial Vehicle

Each of these is described in greater detail in Appendix B. Each performs various operational activities, has communications sub-systems, and performance data as outlined in Figure 44. For this example, this data was stored in a MATLAB structure.

		S1 - Afghan Artillery	S2 - US Artillery	S3 - Afghan HQ	S4 - US HQ	S5 - US Rifle PLT	S6 - US SOF Team	S7 - Afghan Rifle PLT 1	S8 - Afghan Rifle PLT 2	S9 - US UAV
Capability	Shoot	X	X							
	Deconflict			X	X					
	Observe					X	X	X	X	X
Communication Sub-Systems	Afghan FM	X		X	X*		X	X	X	
	OSRTV				X					X
	US FM		X		X	X	X			
	BFT		X		X	X	X			
	MIRC		X		X		X			X
Performance Data	P_{hit}	0.5	0.9	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	P_{kill}	0.85	0.85	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	Memory	1	2	2	5	2	2	1	1	0
	P_{detect}	N/A	N/A	N/A	N/A	0.25	0.25	0.15	0.15	0.15
	P_{locate}	N/A	N/A	N/A	N/A	0.8	0.8	0.33	0.33	0.95
	$P_{classify}$	N/A	N/A	N/A	N/A	0.7	0.8	0.85	0.85	0.5
	Max CFF	N/A	N/A	N/A	N/A	2	2	1	1	2

Figure 44. SoS IDF Example Constituent System Data

The three operational activities and their associated performance measures are:

(1) Shoot

- Definition: To propel a projectile from one location to another.
- Measure 1: Probability of a Hit (P_{hit}) is the probability that a system will hit the location at which it aimed. There is a 1- P_{hit} probability that the fired rounds land at a location other than the one the shooter aimed at.
- Measure 2: Probability of a Kill (P_{kill}) is the probability that a shot fired will kill a target at the location where the round impacts. This is assessed independently for each target at that location.

(2) Deconflict

- Definition: To receive and aggregate information, make a decision about that information to maximize a goal, and then send a message based on this information.
- Measure 1: Memory is a measure of how much information a system can store and process.

(3) Observe

- Definition: To identify a target.
- Measure 1: Probability of detection (P_{detect}) is the probability that a system will detect a target in its field of view.
- Measure 2: Probability of location (P_{locate}) is the probability that a system will correctly identify the location of a target it detected.
- Measure 3: Probability of classification ($P_{classify}$) is the probability that a system will correctly classify a target (as either civilian or enemy).
- Measure 4: Max Call For Fire (Max CFF) is the maximum number of observations a system may make at any given time.

The five communication sub-systems are: "Afghan FM Radio," "One Station Remote Video Terminal" (OSVT), "U.S. FM Radio," "Blue Force Tracker" (BFT), and "My Internet Relay Chat" (MIRC). These are described in further detail in Appendix B. Each communications sub-system has an aggregate performance measure that is the probability that a message sent on that system is received and understood. This is a high level aggregation for more detailed measures such as bandwidth, semantic interoperability, range, and availability. The probabilities are detailed in Table 12. Note that the available communications refactorization is to add an Afghan liaison to the U.S. Headquarters, providing it with Afghan FM capability.

Table 12. Probability Communication System Transmits a Message

Communication System	Probability Message Received and Understood
Afghan FM	80%
OSRVT	85%
U.S. FM	90%
BFT	90%
MIRC	95%

b. Processes

As previously defined, there are three operational activities that the set of potential constituent systems can conduct, "Observe," "Deconflict," and "Shoot." For this example, there are two ways these may be arranged as operational activity flows to achieve the emergent behavior of aimed IDF, either: "Observe then Shoot" or "Observe then Deconflict then Shoot" as seen in Figure 45.

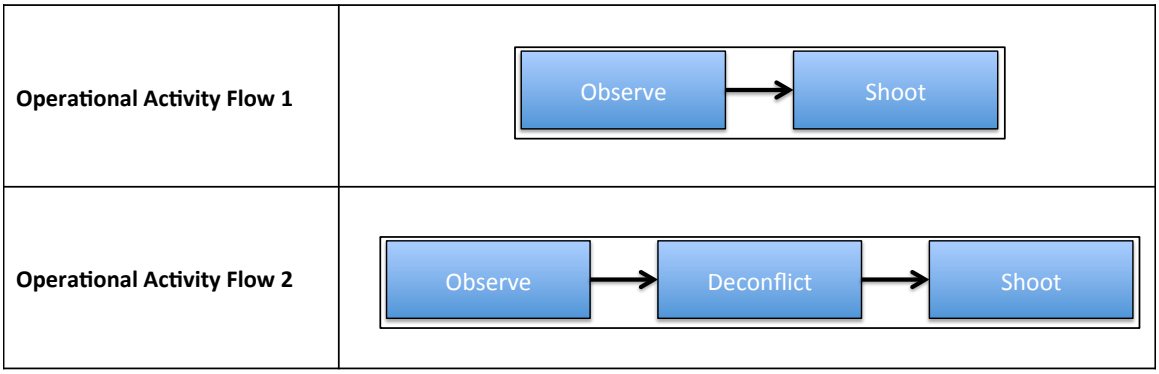


Figure 45. IDF-SoS Operational Activity Flows

Furthermore, there are two sets of rules of employment that define the process parameters. The first concerns the number of independent observations required before shooting—either one or two (e.g., if it is two, both the "UAV" and "Afghan Rifle Platoon" must observe the same target prior to a shooter engaging that target). The second rule concerns how decisions are made regarding which targets to engage, the rules of engagement. The first rule is that a shot cannot be made if civilians are known to be at a location, thus shots are fired at the location with the most enemy but no civilians. The second rule is that shots are chosen to maximize the difference between the number of enemy and civilian targets at a location (e.g., if there are five enemy at a location and two civilians at the same location, the difference is three; if there are two enemy and no civilians at a location, the difference is two; in this case the former location would be engaged).

Taken together, these form eight distinct process architectures:

- P1a: Observe (two independent) → Deconflict → Shoot; Do not engage locations with civilians.
- P1b: Observe (two independent) → Deconflict → Shoot; Maximize enemy – civilian targets at a location
- P2a: Observe → Deconflict → Shoot; Do not engage locations with civilians.
- P2b: Observe → Deconflict → Shoot; Maximize enemy–civilian targets at a location
- P3a: Observe (two independent) → Shoot; Do not engage locations with civilians.
- P3b: Observe (two independent) → Shoot; Maximize enemy–civilian targets at a location.
- P4a: Observe → Shoot; Do not engage locations with civilians.
- P4b: Observe → Shoot; Maximize enemy–civilian targets at a location

c. Organizations

Finally, the organizational inputs to the SoS-TDM for the IDF-SoS are defined through three organizational relationships: "No Relationship," "Collaborative Relationship," and a "Command-Subordinate Relationship."

The first is no relationship. The name of this relationship defines it; two systems with no relationship, even if they *can* communicate, do not communicate. This sort of relationship is beneficial in cases where centralized control is desired; furthermore, it minimizes the amount of interactions that must be accounted (and paid for) during the training and employment of the SoS.

The second relationship is collaborative. Two systems with a collaborative relationship may share information and request activity from each other (e.g., an observer may request fires from a shooter, or a system may request another system pass information along); however, neither system exerts control over the other. Any positive response to a request is purely at the discretion of the requested system, which may prioritize requests as it wishes.

The final relationship is a command-subordinate relationship. In this relationship, the subordinate prioritizes requests from the commander and prioritizes sending information to the commander. This relationship must be used judiciously, as if one system has multiple commanders, the benefit of being a commander is diminished. Furthermore, this relationship may or may not be amenable to all systems.

There is a set of organizational relationships that are acceptable to each system. This is defined in a matrix in which each system is defined along the rows and columns of the matrix as seen in Table 13. The *i-j* cell of the matrix defines the relationship as *i* is the ___ of *j* (e.g., *i* is the subordinate of *j*). If a relationship is found in the *i-j* cell, then that relationship is acceptable to the i^{th} system. The abbreviations are "Cmd" for command, "Sub" for subordinate, "Col" for collaborative, and "N" for no relationship.

Table 13. Table of Acceptable Organizational Relationships

	Afghan Artillery	U.S. Artillery	Afghan HQ	U.S. HQ	U.S. Rifle Platoon	U.S. Special Operations Team	Afghan Rifle Platoon - 1	Afghan Rifle Platoon - 2	U.S. UAV
Afghan Artillery	-	Cmd, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Col, N	Cmd, Col, N	Cmd, Sub, Col, N
U.S. Artillery	Cmd, Col, N	-	Cmd, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Col, N	Cmd, Col, N	Cmd, Sub, Col, N
Afghan HQ	Cmd, Col, N	Cmd, Col, N	-	Cmd, Sub, Col, N	Cmd, Col, N	Cmd, Sub, Col, N	Cmd, Col, N	Cmd, Col, N	Cmd, Col, N
U.S. HQ	Cmd, Col, N	Cmd, Col, N	Cmd, Col, N	-	Cmd, Col, N	Cmd, Col, N	Cmd, Col, N	Cmd, Col, N	Cmd, Col, N
U.S. Rifle Platoon	Cmd, Col, N	Cmd, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	-	Cmd, Sub, Col, N	Cmd, Col, N	Cmd, Col, N	Cmd, Col, N
U.S. Special Operations Team	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	-	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N
Afghan Rifle Platoon - 1	Cmd, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	-	Cmd, Col, N	Cmd, Sub, Col, N
Afghan Rifle Platoon - 2	Cmd, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Col, N	-	Cmd, Sub, Col, N
U.S. UAV	N	Cmd, Sub, Col, N	N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	N	N	-
Acceptable Relationships: Cmd = Command; Sub = Subordinate; Col = Collaborative; N = No Relationship									

Finally, there are 11 ways in which these relationships define distinct organizations. Each is depicted visually in which a system is a blue block, a collaborative relationship is depicted as a black line, and a command-subordinate relationship is depicted as a black arrow in which the arrow points from the commander to the subordinate. A non-relationship is simply not indicated. There are red lines and arrows that indicates the relationship depends upon the inclusion of the Afghan LNO

refactorization to the U.S. Headquarters. Note that these images are small; larger versions are included in Appendix B.

The first set of organizations is based upon likely existing hierarchies—by country and by command; they are further modified by allowing collaboration at the subordinate level (i.e., two subordinates of the same commander may interact or not). These hierarchy variations form the first six possible organizations.

Organizations 1a and 1b are formed from collaboration between a hierarchy of U.S. and Afghan forces. The distinguishing characteristic between the two is that in the first, no collaboration is allowed among subordinates and, in the second, it is. These are seen in Figure 46.

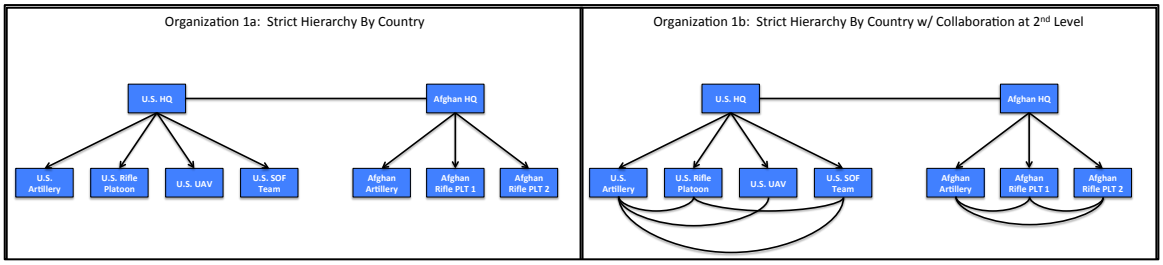


Figure 46. Organizations 1a and 1b

The second set of organizations is a modification of the Organizations 1a and 1b in which the U.S. Headquarters commands the Afghan Headquarters, and, by extension, all Afghan forces.

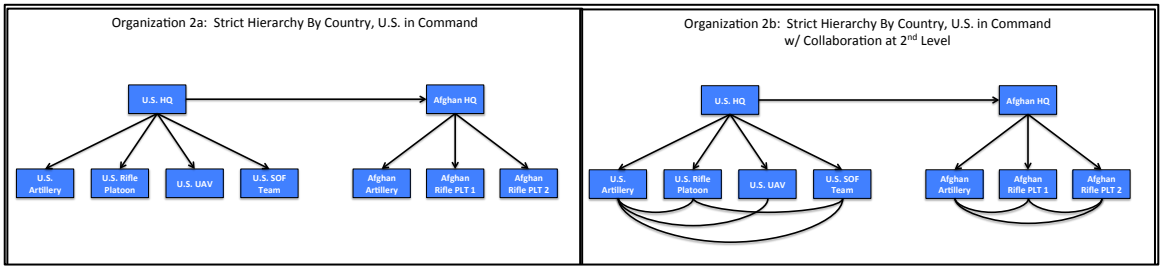


Figure 47. Organizations 2a and 2b

The third set is the reverse of Organizations 2a and 2b with the Afghan Headquarters commanding the U.S. Headquarters.

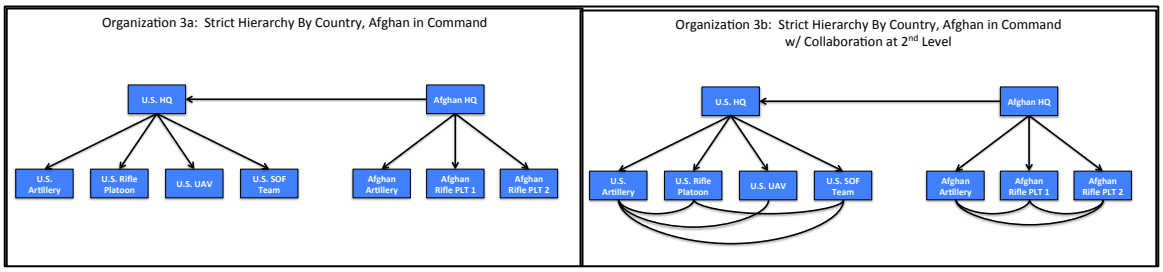


Figure 48. Organizations 3a and 3b

The fourth set of organizations is those that are arranged by command. At the top level, the various commands—the U.S. Army, U.S. Air Force, U.S. Special Operations, and Afghan Army—are all collaborative.

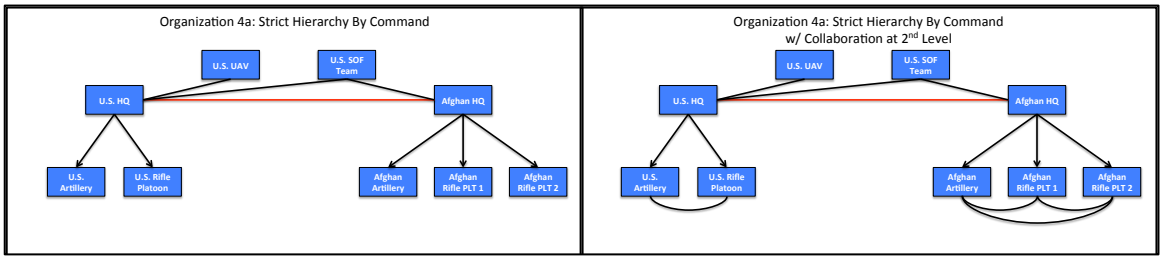


Figure 49. Organizations 4a and 4b

Organization 5 is different from the previous hierarchical organizations; it is a purely collaborative organization (up to communication ability). Any two systems that *can* communicate *may* communicate.

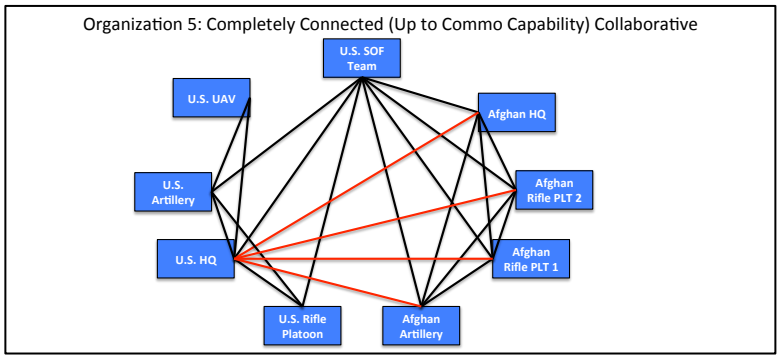


Figure 50. Organization 5

The final set of organizations is functionally based. The organizational relationships are defined based upon the functions each system performs. In the first, the headquarters (which perform the deconflict function) are central and can interface with any shooter system or observer system. The shooters may all collaborate and the observers may all collaborate (up to communication). The second is similar, except it only considers observers and shooters.

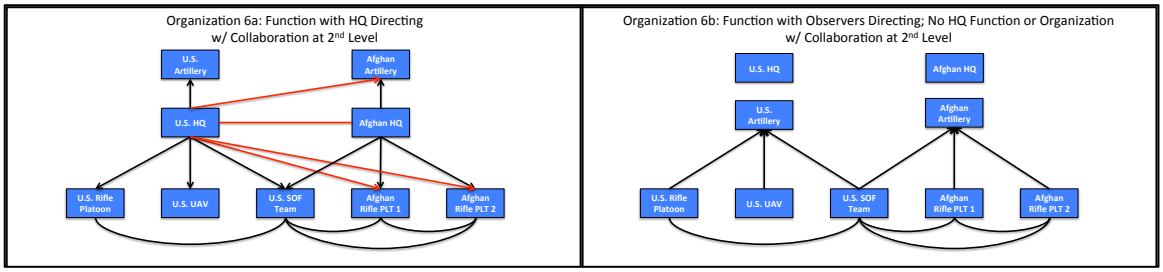


Figure 51. Organizations 6a and 6b

Collectively, these inputs: the physical—the systems and their communications sub-systems, the processes—the operational activity flows and rules of employment, and the organization—the relationships and their potential structure define the second major input to the SoS-TDM. Combined with the first input, the valid SoS need and performance measures, these inputs allow the use of the SoS-TDM.

B. IDF SOS-TDM STEP 1: IDF DESIGN SPACE DEFINITION

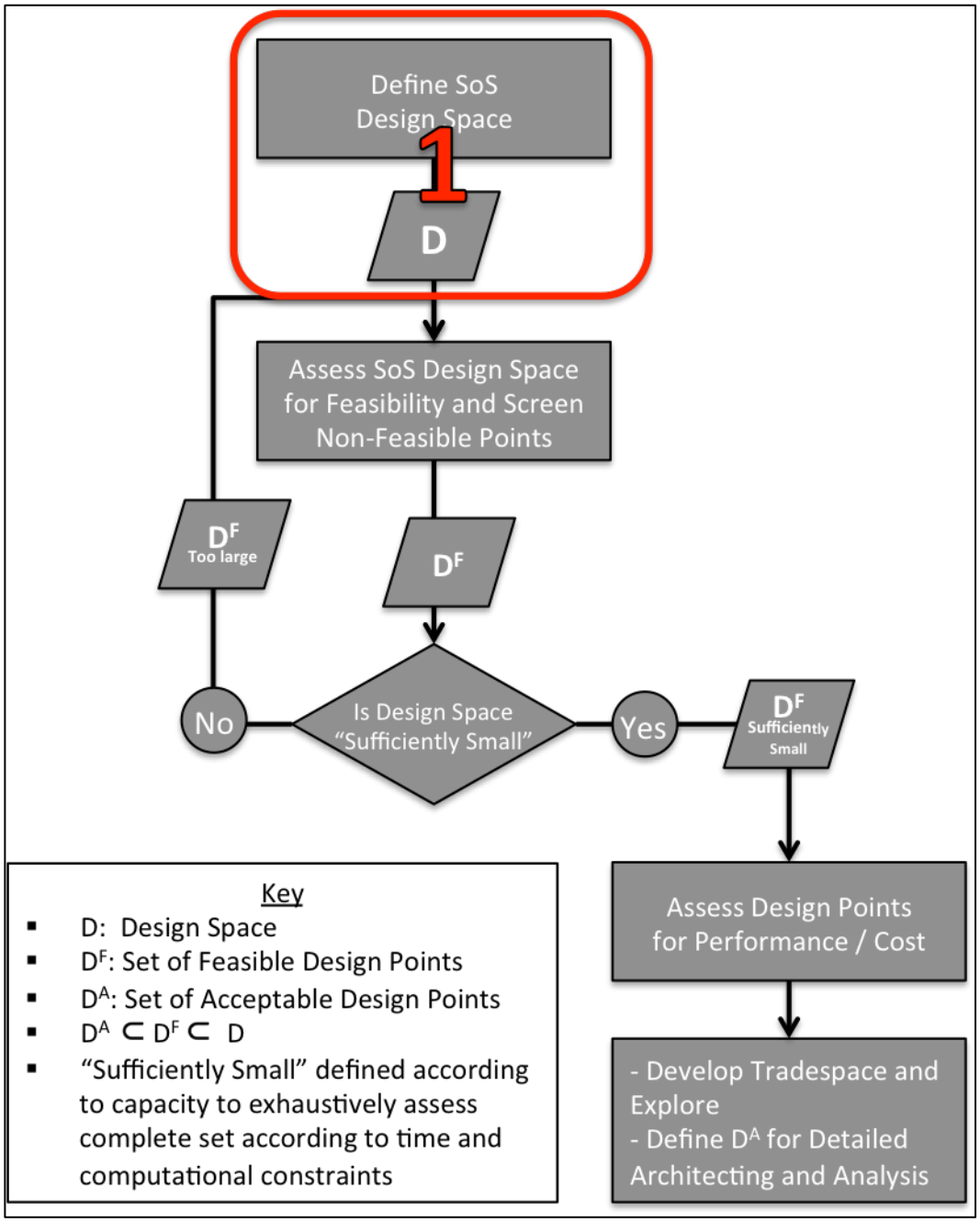


Figure 52. SOS-TDM – Define SoS Design Space

The first step of the SoS-TDM is to define the design space as depicted in Figure 52. Recall that the design space is defined as the set of points defined by the Cartesian product of the physical, process, and organization design spaces: $\boldsymbol{D = D^{Phys} \times D^{Proc} \times}$

$\boldsymbol{D^{Org}}$; each of these sub-spaces is defined by the Cartesian product of the domain for each parameter. For this problem, this leads to a 1x14 vector in which each entry is a variable defining some aspect of this space as seen below and described in Table 14.

$$\langle S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, F_1, E_1, E_2, O \rangle$$

Table 14. Design Space Parameter Definition and Domains

Parameter	Description	Domain	Magnitude of the Domain
S_1	System 1 – Afghan Artillery	[0, 1]	2
S_2	System 2 – U.S. Artillery	[0, 1]	2
S_3	System 3 – Afghan Headquarters	[0, 1]	2
S_4	System 4 – U.S. Headquarters	[0, 1]	2
S_5	System 5 – U.S. Rifle Platoon	[0, 1]	2
S_6	System 6 – U.S. Special Ops. Team	[0, 1]	2
S_7	System 7 – Afghan Rifle PLT – 1	[0, 1]	2
S_8	System 8 – Afghan Rifle PLT – 2	[0, 1]	2
S_9	System 9 – U.S. UAV	[0, 1]	2
S_{10}	Afghan Liaison	[0, 1]	2
F_1	Operational Activity Flow	[Operational Activity Flow 1, Operational Activity Flow 2]	2
E_1	Number of Required Observers	[1, 2]	2
E_2	Rules of Engagement	[No Civilians (1), Max Difference Enemy – Civilian (-1)]	2
O	Organization	[Org 1a, Org 1b, Org 2a, Org 2b, Org 3a, Org 3b, Org 4a, Org 4b, Org 5, Org 6a, Org 6b]	11

These variables and definitions correspond exactly to the inputs as described in the previous section; this merely formalizes the inputs into the notation described in Chapter III. From this, we can easily see that the magnitude of the design space, the product of the magnitude of each parameter's domain, is:

$$2\times2\times2\times2\times2\times2\times2\times2\times2\times2\times2\times2\times11 = 90{,}112$$

Finally, note that there is significant interplay in defining the inputs for the SoS-TDM and this first step. In the input section, we defined each system, process, and organization specifically to align with the requirements for the SoS-TDM Step 1 as defined in Chapter III. For clarity, those inputs were defined in the previous section and formalized in this section; however, in reality, this distinction is blurred and necessarily iterative.

C. IDF SOS-TDM STEP 2: IDF DESIGN SPACE FEASIBILITY ANALYSIS AND SCREENING: THE SOS-AFAM

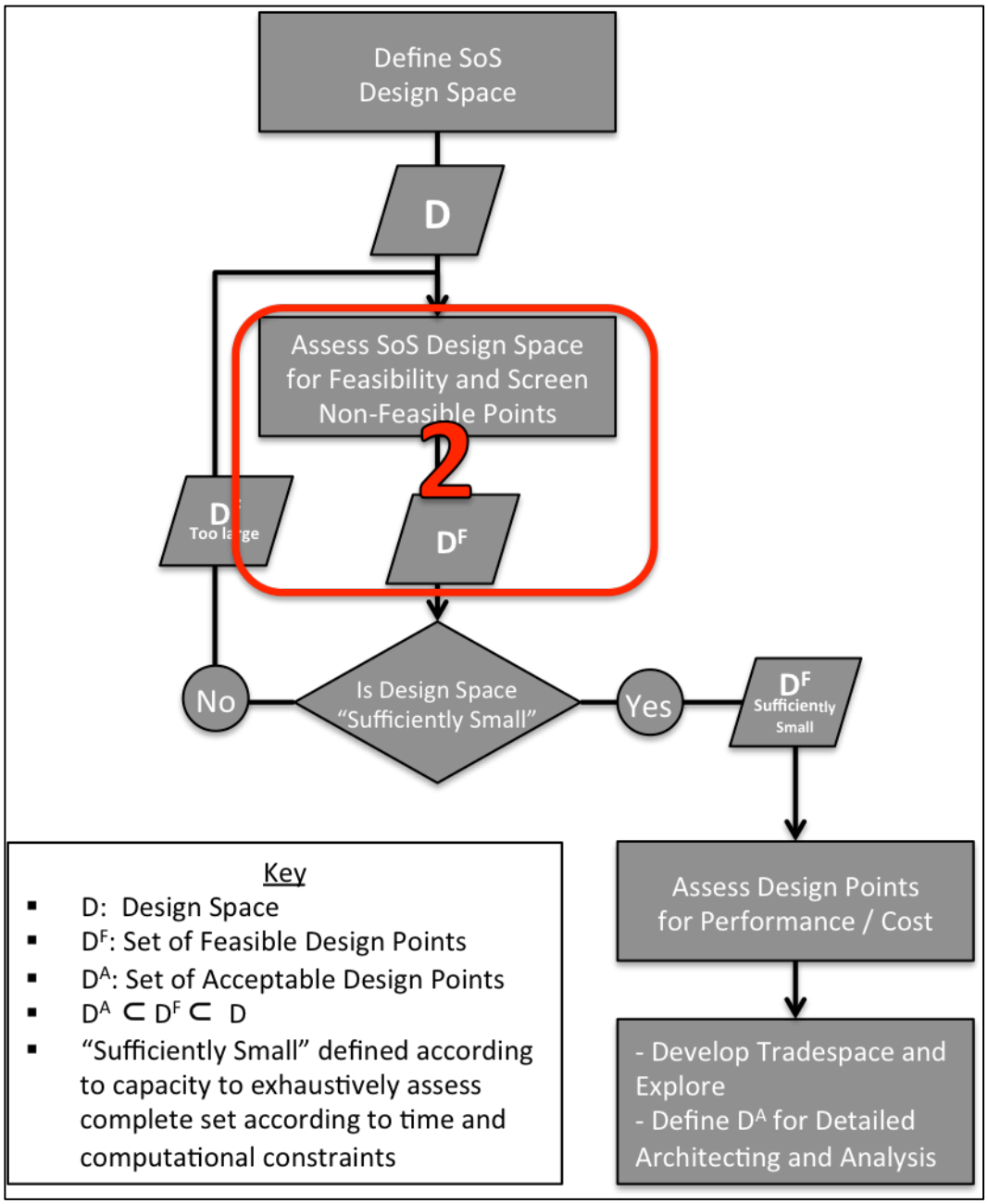


Figure 53. SoS-TDM – Design Space Feasibility Analysis and Screening

The second step of the SoS-TDM for the IDF-SoS is to assess the IDF design space as seen in Figure 53. This is done through the four steps of the SoS-AFAM and described in the next four sections.

1. IDF SoS-AFAM Step 1: IDF Physical Design Space Feasibility Analysis

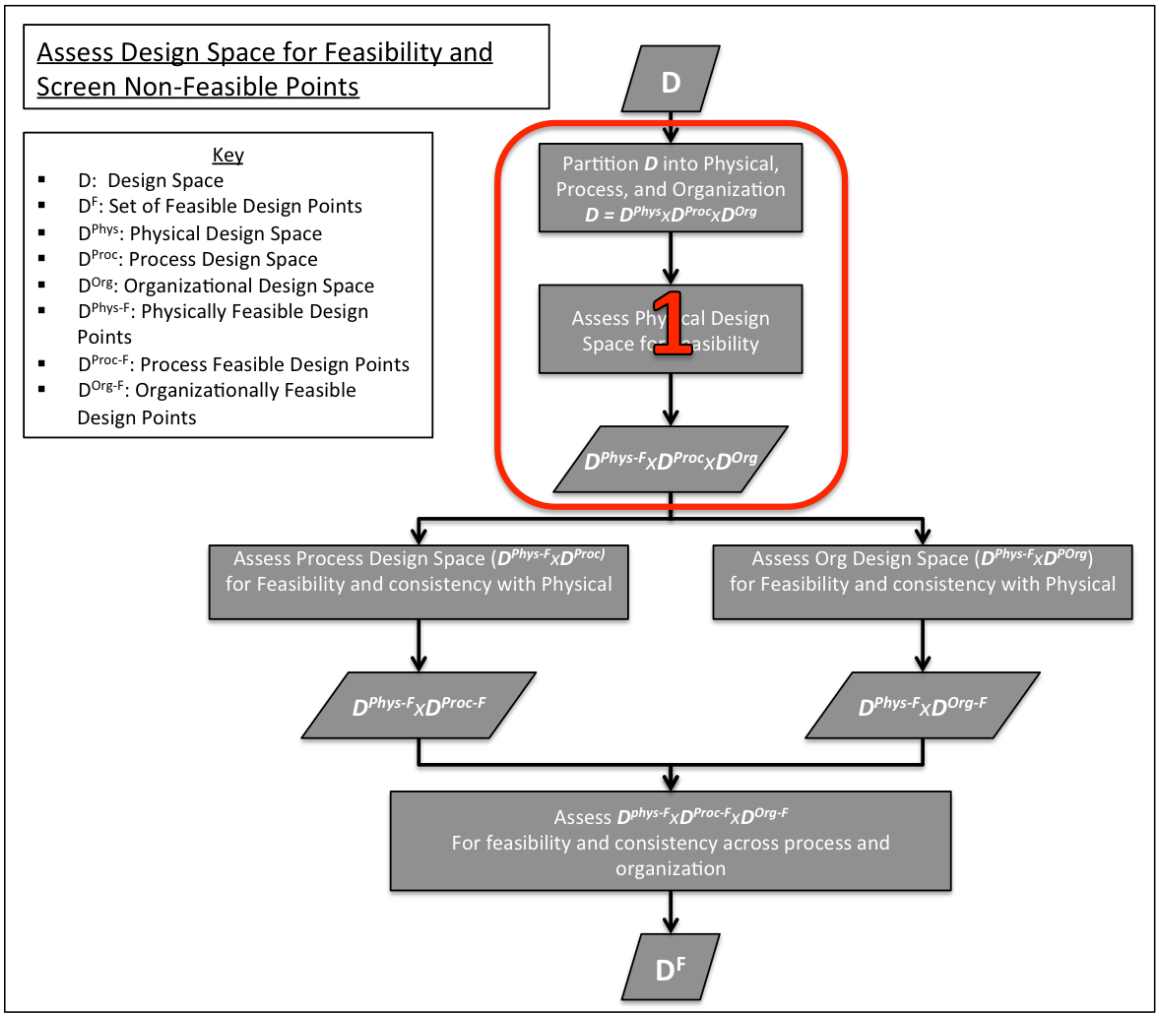


Figure 54. SoS-AFAM Step 1: Physical Design Space Feasibility Analysis

The first step of the SoS-AFAM is to assess the physical portion of the design space for feasibility as depicted in Figure 54. For the IDF-SoS, this involves assessing which compositions of systems form a connected network.

The initial physical SoS feasibility defined two systems in a given physical design point as connected if they shared a common communications sub-system as defined in Figure 44. To do this, we implemented Algorithm 1 as defined in Section III.C.1. Note that this algorithm checks every design point of the form: $\langle S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10} \rangle$ where each variable is binary. This results in

$2^{10} = 1{,}024$ physical design points. It assesses each combination of these variables against a 9x9 matrix whose entries are one if the corresponding systems share any communications means and a zero otherwise. This matrix is seen in Table 15. Finally, note that the algorithm was also modified to exclude any potential compositions of SoS that only contained one or zero systems.

Table 15. Initial System-System Connectivity Matrix

	Afghan Artillery	U.S. Artillery	Afghan H.Q.	U.S. H.Q.	U.S. Rifle PLT	SOF Team	Afghan Rifle PLT - 1	Afghan Rifle PLT - 2	UAV
Afghan Artillery	0	0	1	0	0	1	1	1	0
U.S. Artillery	0	0	0	1	1	1	0	0	1
Afghan H.Q.	1	0	0	0	0	1	1	1	0
U.S. H.Q.	0	1	0	0	1	1	0	0	1
U.S. Rifle PLT	0	1	0	1	0	1	0	0	0
SOF Team	1	1	1	1	1	0	1	1	0
Afghan Rifle PLT - 1	1	0	1	0	0	1	0	1	0
Afghan Rifle PLT - 2	1	0	1	0	0	1	1	0	0
UAV	0	1	0	1	0	0	0	0	0

The result of this analysis is that there are 372 physically connected combinations of systems of the 1,024 potential ones. The time to run this analysis was negligible, only a second or two, but it provided a 64% reduction in the design space. Note that this does not, in and of itself, mean that any of those 372 physical compositions forms an entirely feasible SoS. For example, the SoS composed of the "Afghan Rifle Platoon" and "SOF Team" meets the requirement of forming a connected network; however, it clearly is not a viable SoS as it has no ability to shoot. Another example, the SoS formed by the "U.S. Rifle Platoon" and "U.S. Artillery" may be a viable SoS, but it depends upon the required process; if the operational activity flow is "Observe" then "Shoot," then it is may be a viable SoS; if the activity flow is "Observe," then "Deconflict," then "Shoot," then it is not a viable SoS. These questions are addressed in the process, organization, and total design space feasibility analysis sections.

For this analysis, the 64% reduction in the physical design space was, in combination with the subsequent process, organization, and total analyses, sufficient to winnow the feasible design space for exhaustive operational and cost analysis. For demonstration, however, we assessed each of the 372 initially feasible design points against the probability that they formed a connected network when the communications sub-systems had a probability of not correctly passing a message according to Table 12. This involved a slight modification of Algorithm 1. To do this, we assessed each of the 372 design points 100 times. Each time, the connectivity matrix, Table 15, was modified so that connectivity between two systems was dependent upon the probability that each communications device worked. For example, the sole connection between the "Afghan H.Q." and the "Afghan Artillery" is through the "Afghan FM." This has an 80% chance of correctly connecting and sending a message. Therefore, 20% of the time the "Afghan H.Q." and "Afghan Artillery" systems were not connected. From there, we executed Algorithm 1 as defined. This took approximately one minute to execute and the result of this is that some of the 372 initially feasible SoS did not always form a connected network as seen in Figure 55.

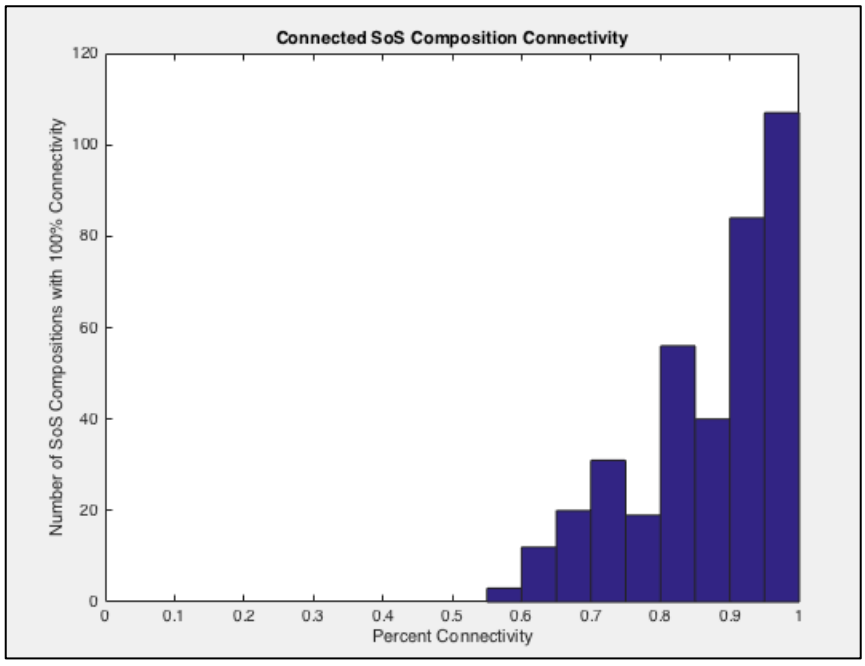


Figure 55. SoS Composition Likelihood of Connectivity

2. IDF SoS-AFAM Step 2: IDF Process Design Space Feasibility Analysis

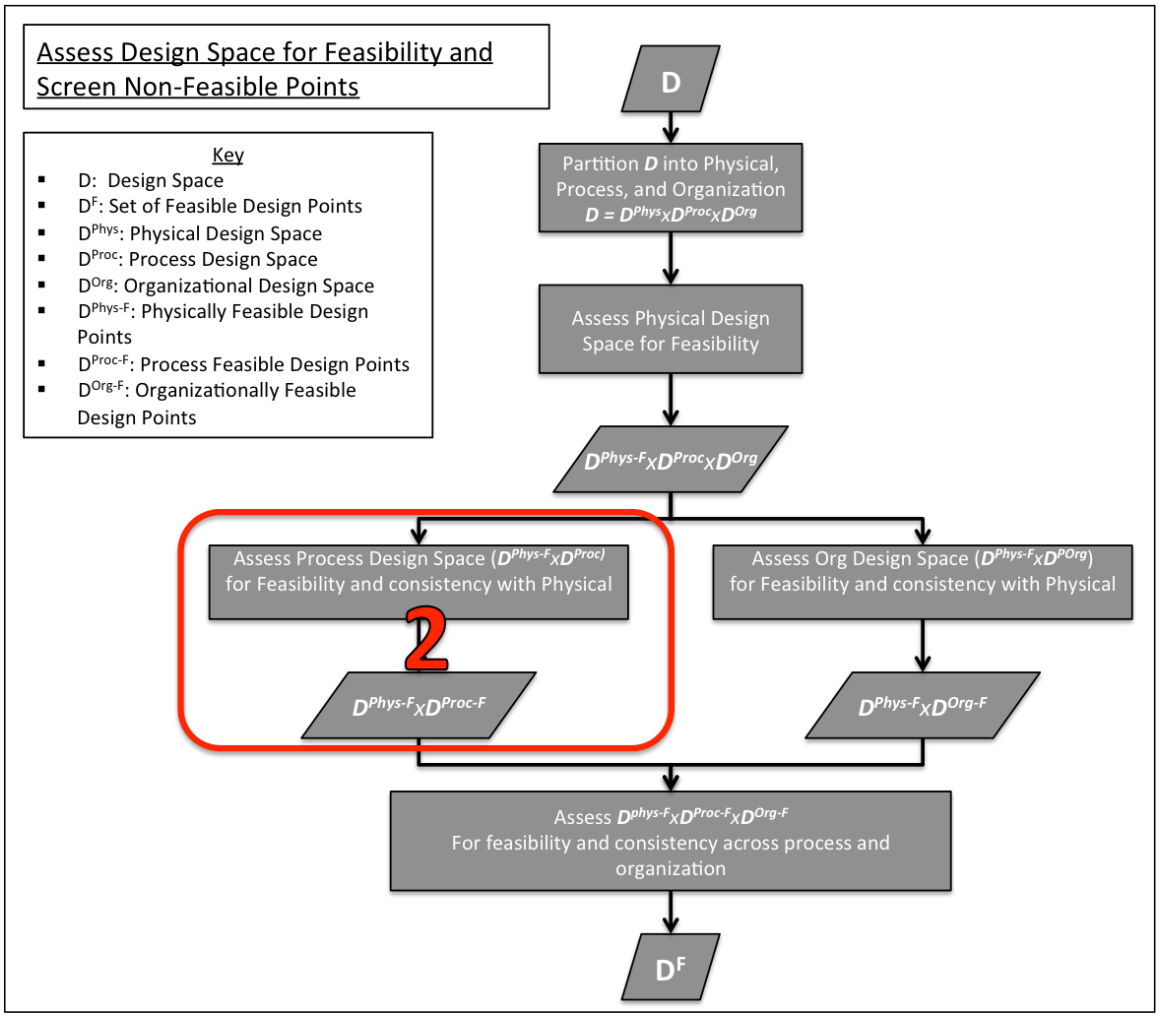


Figure 56. SoS-AFAM Step 2: Process Design Space Feasibility Analysis

The second step of the SoS-AFAM is to assess the process design space for feasibility as depicted in Figure 56. The input is the set of feasible physical design points crossed with the set of all possible processes; the size of this is 372×2×2×2=2,976 as there are 372 possible feasible compositions of systems, two operational activity flows, and two sets of rules, each with two options. To assess each design point for process viability, we must first define each of the eight distinct processes and the number of each type of function they require. This is depicted in Table 16. Each is numbered for convenience. Note that one of the rules of employment, the number of independent

observations, affects the number of required functionalities whereas the other rule does not affect the functionality requirement.

Table 16. IDF-SoS Processes versus Required System Functionality

Process #	**F1: Operational Activity Flow**	**E1: Number of Independent Observations**	**E2: Rules of Engagement**	**Min # Observer Systems Required**	**Min # Deconflicter Systems Required**	**Min # Shooters Required**
1a	Observe → Deconflict → Shoot	1	No Civilian	1	1	1
1b	Observe → Deconflict → Shoot	1	Max Difference	1	1	1
2a	Observe → Deconflict → Shoot	2	No Civilian	2	1	1
2b	Observe → Deconflict → Shoot	2	Max Difference	2	1	1
3a	Observe → Shoot	1	No Civilian	1	0	1
3b	Observe → Shoot	1	Max Difference	1	0	1
4a	Observe → Shoot	2	No Civilian	2	0	
4b	Observe → Shoot	2	Max Difference	2	0	1

The functionality of each system is depicted in Figure 44. Taking the functionality provided by each system combined with the minimum requisite functionality for each process, one may assess if a set of systems is process feasible by using Algorithm 2 defined in Section III.C.2. The number of feasible systems for each process is depicted in Table 17. Note that the rule of employment does not impact functionality test, although it could impact subsequent testing if required.

Table 17. Number of Feasible SoS by Process

Process	1a	1b	2a	2b	3a	3b	4a	4b
# Feasible SoS	207	207	235	235	246	246	281	281

The end result is that 1,938 physical-process design points are feasible from a process perspective. This is a 35% reduction from the initial set of 2,976 potential design points. The run time of this analysis was negligible, only a second or two.

Although it was not necessary in this example, one could further prune this design space in two ways. The first would be by assessing acceptance of the rules of employment in a given process against the systems included in the SoS. This would be done by defining a matrix of system acceptance or non-acceptance of each rule as depicted in Table 7. The second more detailed assessment is to identify process conflicts and identify SoS that contain these conflicts as described in Algorithm 3 and depicted in Table 8.

3. IDF SoS-AFAM Step 3: IDF Organization Space Feasibility Analysis

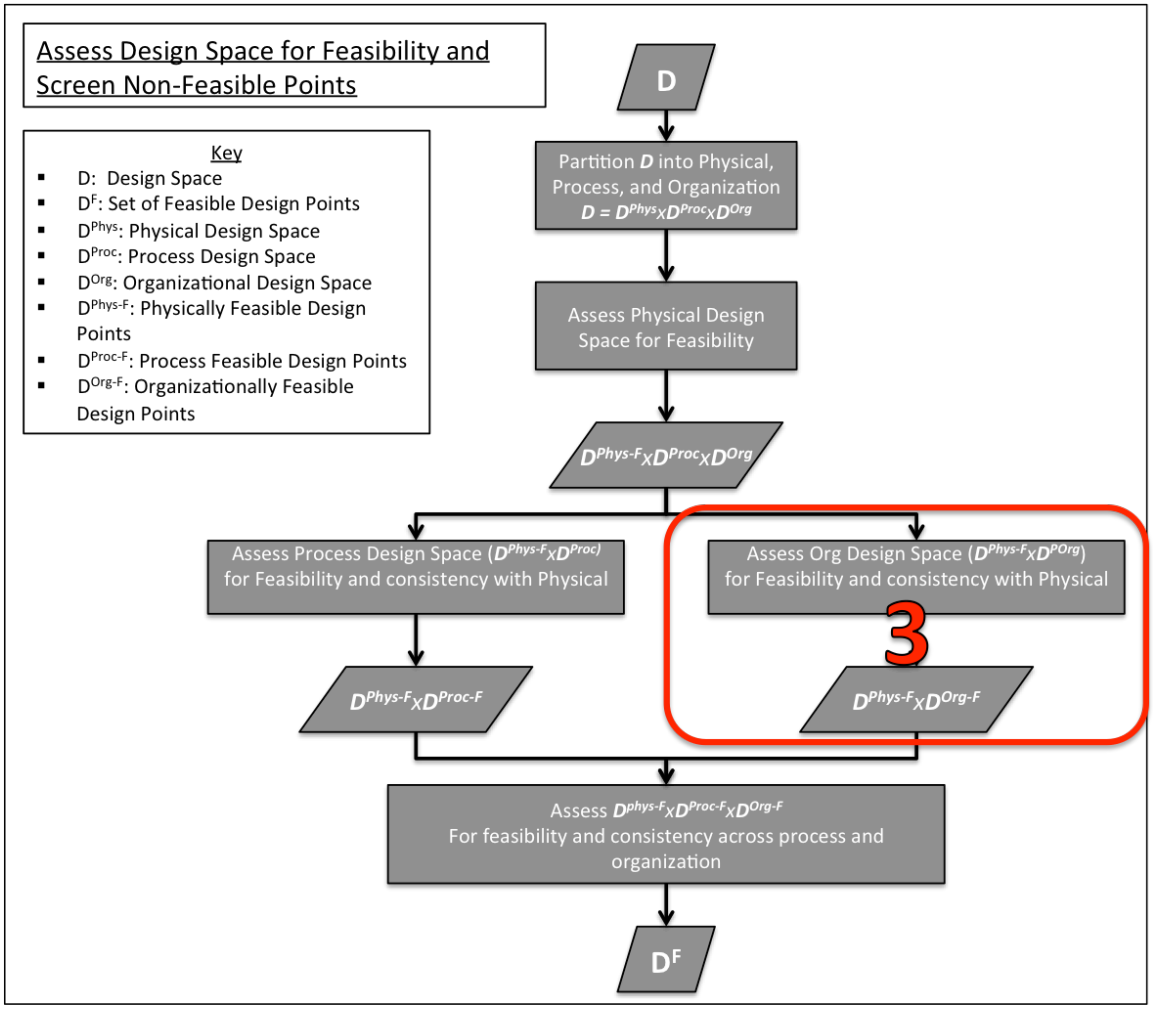


Figure 57. SoS-AFAM Step 3: Organization Design Space Feasibility Analysis

The third step of the SoS-AFAM is to assess the organization design space for feasibility as depicted in Figure 57. The design points tested in this section are the cross between the 372 physically feasible design points and the 11 possible organizations, for a total of 4,092 design points. This feasibility assessment is done through a series of three tests.

The first test is to assess if a given organization is acceptable to a set of systems. This is done by comparing the organization matrix for the set of systems relative to the set of acceptable relationships as defined by Algorithm 4 in Section III.C.3. The organization matrix for a set of systems is simply the organization matrix for that organization design modified so that it only represents the systems included in that design point. This is simply checked against the system-system organizational relationship matrix. If a design point contains a single non-acceptable organizational relationship it is deemed infeasible. For example, the SoS with all nine systems and Organization 3a or 3b is infeasible as it has the "Afghan Headquarters" in command of the "U.S. Headquarters" and this relationship is not acceptable to the "U.S. Headquarters" by the set of acceptable relationships in Table 13.

The second test assesses the set of design points that were determined to be feasible from an organizational acceptance perspective against the connectivity requirement. In this requirement, we assess each organizational matrix for connectedness. This is done through Algorithm 5 in Section III.C.3. The input for each design point to be assessed is an adjacency matrix in which two systems have a common link if they have an organizational relationship.

Finally, the last test takes the previous results, and assesses if each relationship in the design point is supported by a physical communications sub-system as defined by Algorithm 7A in Section III.C.3. That is, if the "U.S. Headquarters" and "U.S. Artillery" are included in the system and the "U.S. Headquarters" commands the "U.S. Artillery," this is feasible as these two systems share a common communications sub-system, "U.S. FM." On the other hand, if the "U.S. Headquarters" commands the "Afghan Artillery," and the "Afghan Liaison" is not present, this is not feasible as the two systems do not share a common communications sub-system.

The end result of these three tests was that 1,677 of the 4,092 possible design points were found organizationally feasible as depicted in Table 18. This is a 59%

reduction in the set of physical-organization design points. The run time for these analyses was similarly quick as the physical and process ones, only a second or two. Note that the exceptionally low results in 6b are due to the fact that some systems are inherently excluded in 6b (the deconflicters, the "U.S. Headquarters" and "Afghan Headquarters"). Furthermore, there are significant symmetries in the first three organizations that lead to similar results.

Table 18. Results of Organization Architecture Analysis

Organization	1a	1b	2a	2b	3a	3b	4a	4b	5	6a	6b
# Feasible	150	105	150	105	150	105	222	227	259	150	54

4. IDF SoS-AFAM Step 4: Total IDF Design Space Feasibility Analysis

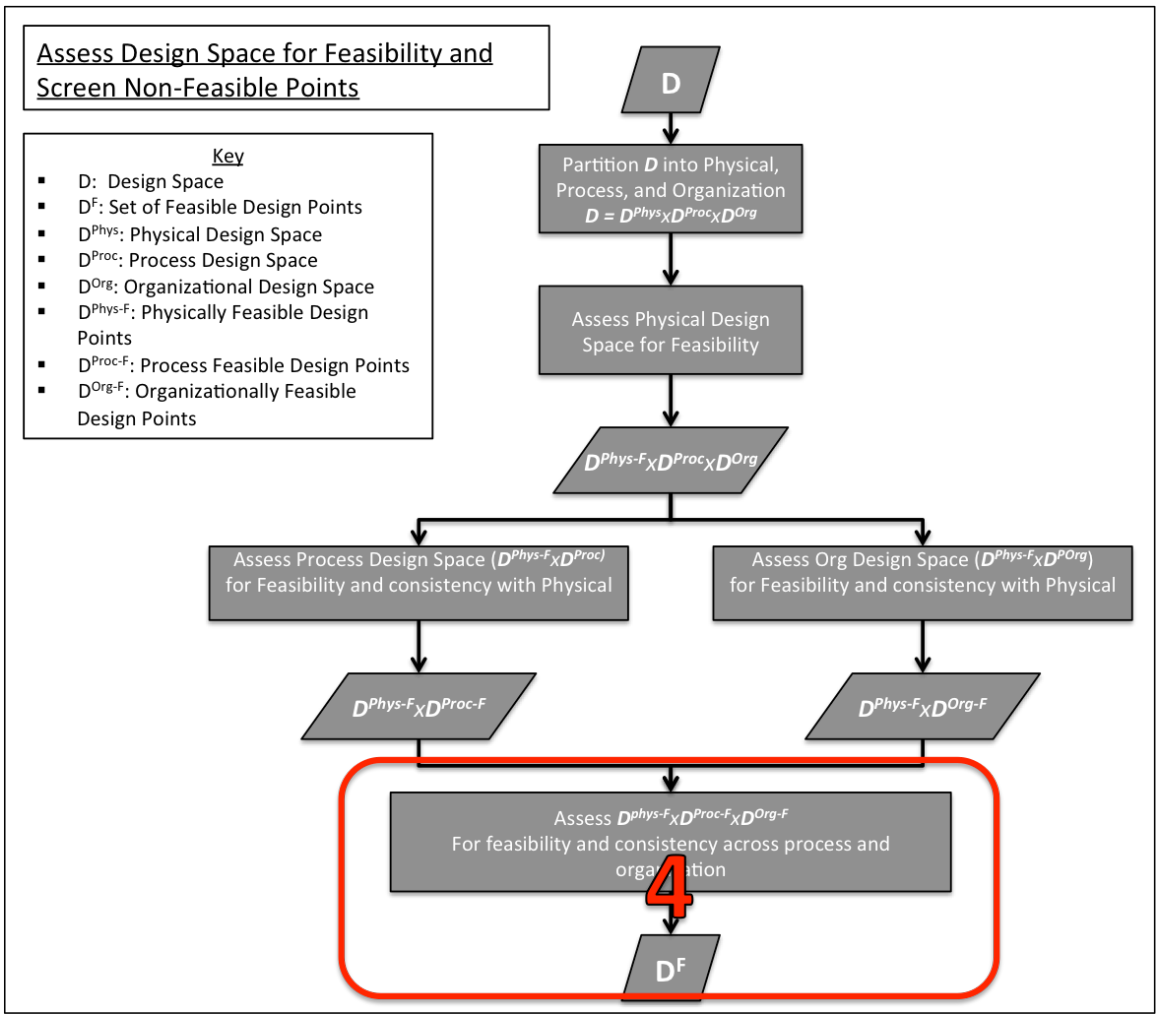


Figure 58. SoS-AFAM Step 4: Total Design Space Feasibility Analysis

The final check for the IDF-SoS is to assess the design space in its totality. Recall that initially there were 90,112 design points defined by the physical, process, and organizational parameters. We have identified 1,938 process and physically feasible points and 1,677 organizationally and physically feasible points, each defined by a composition of systems and a process or organization (e.g., the "U.S. Artillery," "U.S. Headquarters" and "U.S. Rifle Platoon" with the "Observe," "Deconflict," "Shoot," one observer, and no civilian present physical-process design point). To be totally feasible, a design point, defined by its physical composition, process, and organization, must be

feasible against each one of these perspectives. Furthermore, the organization must support the process requirements.

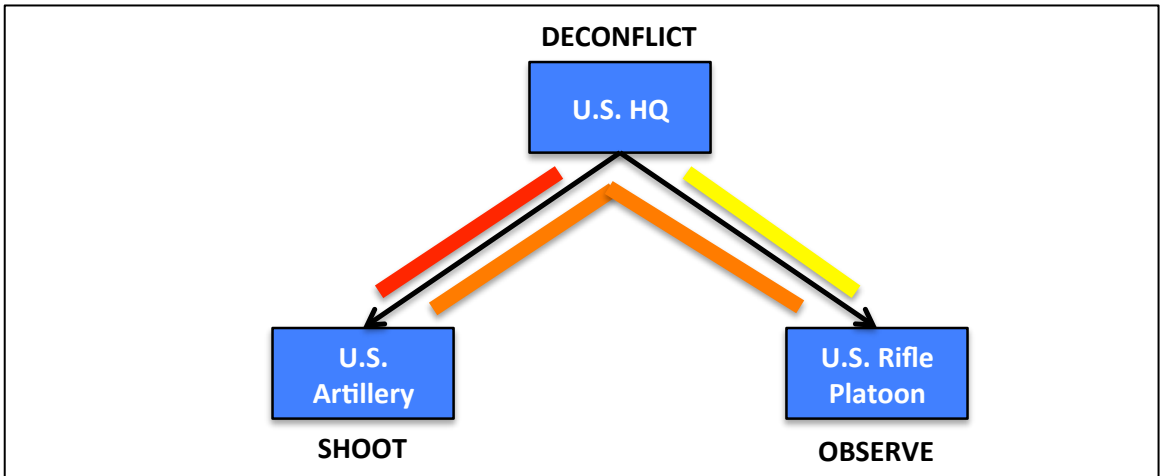
The first check is to identify the points that are physically, organizationally, and process feasible as defined in Algorithm 8 in Section III.C.4. In general, we already have a defined set of physical-organization design points that are feasible and a set of physical-process design points that are feasible. We only must check either the feasible physical-organization design points crossed with all potential processes or the reverse. For example, in Figure 18 we see that the physical organization design point of the "U.S. Artillery," "U.S. Headquarters," "U.S. Rifle Platoon" and Organization 1a is feasible. Therefore, there are eight potentially feasible physical-organization-process design points: the initial point with each of the eight processes; however, not all of them are totally feasible. We know from the process analysis that some of the processes with this physical design are not feasible.

Table 19. Feasible Physical-Organization Design Point Crossed with All Eight Processes

Physical – Organization	**Process**	**Feasible**	**Note**
(U.S. Artillery, U.S. Headquarters, U.S. Rifle Platoon) x (Organization 1a)	1a	Yes	Has sufficient functionality
	1b	Yes	Has sufficient functionality
	2a	No	Needs an additional "Shooter"
	2b	No	Needs an additional "Shooter"
	3a	Yes	Has sufficient functionality
	3b	Yes	Has sufficient functionality
	4a	No	Needs an additional "Shooter"
	4b	No	Needs an additional "Shooter"

The end result of this analysis is that 7,980 design points are feasible from a physical, organization, and process perspective. The computational time of this analysis was only a second. This resulted in a 76% reduction from checking every possible organization and process against the 372 physically feasible designs (32,736 design points). This reduces the size of the feasible design space to a "sufficiently small" number as defined in the next section. If desired, however, one can conduct more detailed total design space analysis.

The more detailed total design space analysis takes the 7,980 design points and assesses how well the organization supports the process as described in Algorithm 9 of Section III.C.4. If one wishes to limit the number of "organizational steps," one must take between any two points in a given process, we follow the algorithm as described. For example, consider the feasible systems described in Table 19. One sees that either operational activity flow is acceptable, so long as only one observer is required. There are, however, varying numbers of "organizational steps" as depicted in Figure 59. If a decision-maker wished to only allow one organizational step between any two operational activities in an operational activity flow, this would restrict any operational activity flow that required an observer to interact directly with a shooter, namely the first one, "Observe" then "Shoot." Thus, of the four initially feasible design points, only two would be feasible.



OBSERVE → SHOOT	2 x Organizational Steps	
OBSERVE → DECONFLICT	1 x Organizational Steps	
DECONFLICT → SHOOT	1 x Organizational Steps	

Figure 59. Example Number of Organizational Steps for a Design Point

From the 7,980 feasible design points, if one restricts the number of organizational steps between any two necessary operational activities to one, using

Algorithm 9, we reduce the feasible design space to 4,806 design points, a 40% reduction. This analysis only took a few seconds to run.

D. IDF SOS-TDM STEP 3: IDF FEASIBLE DESIGN SPACE ANALYSIS

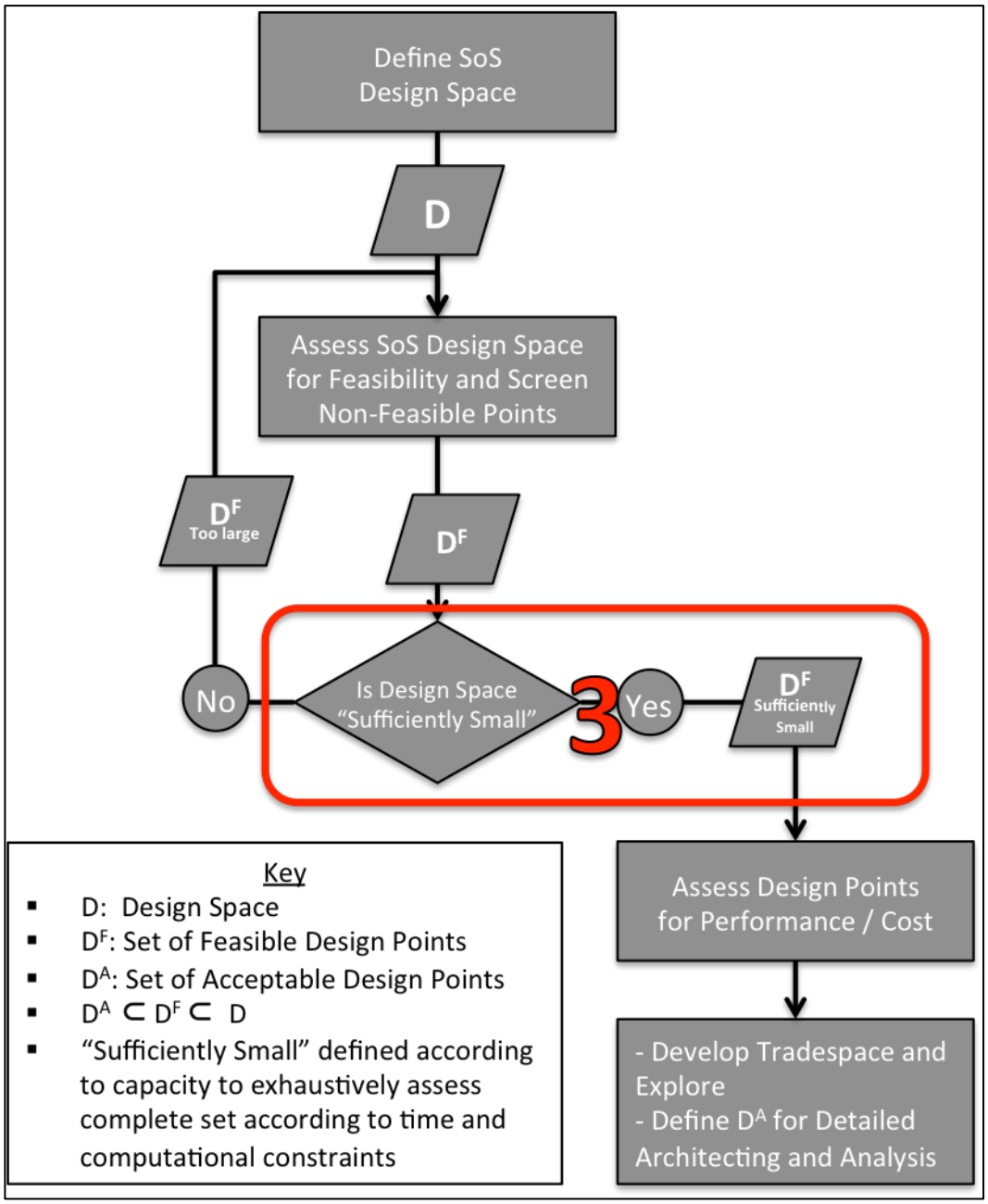


Figure 60. SOS-TDM – Feasible Design Space Analysis

The third step of the SoS-TDM, as seen in Figure 60, is to assess the feasible design space, $\boldsymbol{D}^{\boldsymbol{F}}$, to see if it is "sufficiently small" for exhaustive assessment. If the space is not sufficiently small, one iterates the SoS-AFAM at a higher level of fidelity. For this example, each iteration was combined and discussed in the previous section. The sole question for the IDF-SoS for this section is defining "sufficiently small."

The operational model that assessed PTD and PCD took approximately one minute to run 30 repetitions of a single design point. The cost model took less than a second to assess each design point. For this example, we considered a week of computational time to be the maximum allowable run-time for the operational assessments. Therefore, using the formalization of Section III.D, a design space of less than 10,080 design points is "sufficiently small" as

$$\frac{Time\ Available}{Assessment\ Time\ Per\ Design\ Point} = \frac{1\ Week = 7{\times}24{\times}60 = 10{,}080\ minutes}{1\ minute\ per\ design}$$

The initial design space, with 90,112 design points would require approximately 1,500 hours (two months) of run time to exhaustively assess the entire design space. The feasible design space, with 7,980 design points, is feasible as it can be assessed in less than a week.

E. IDF SOS-TDM STEP 4: IDF DESIGN POINT ASSESSMENT AND TRADESPACE ANALYSIS

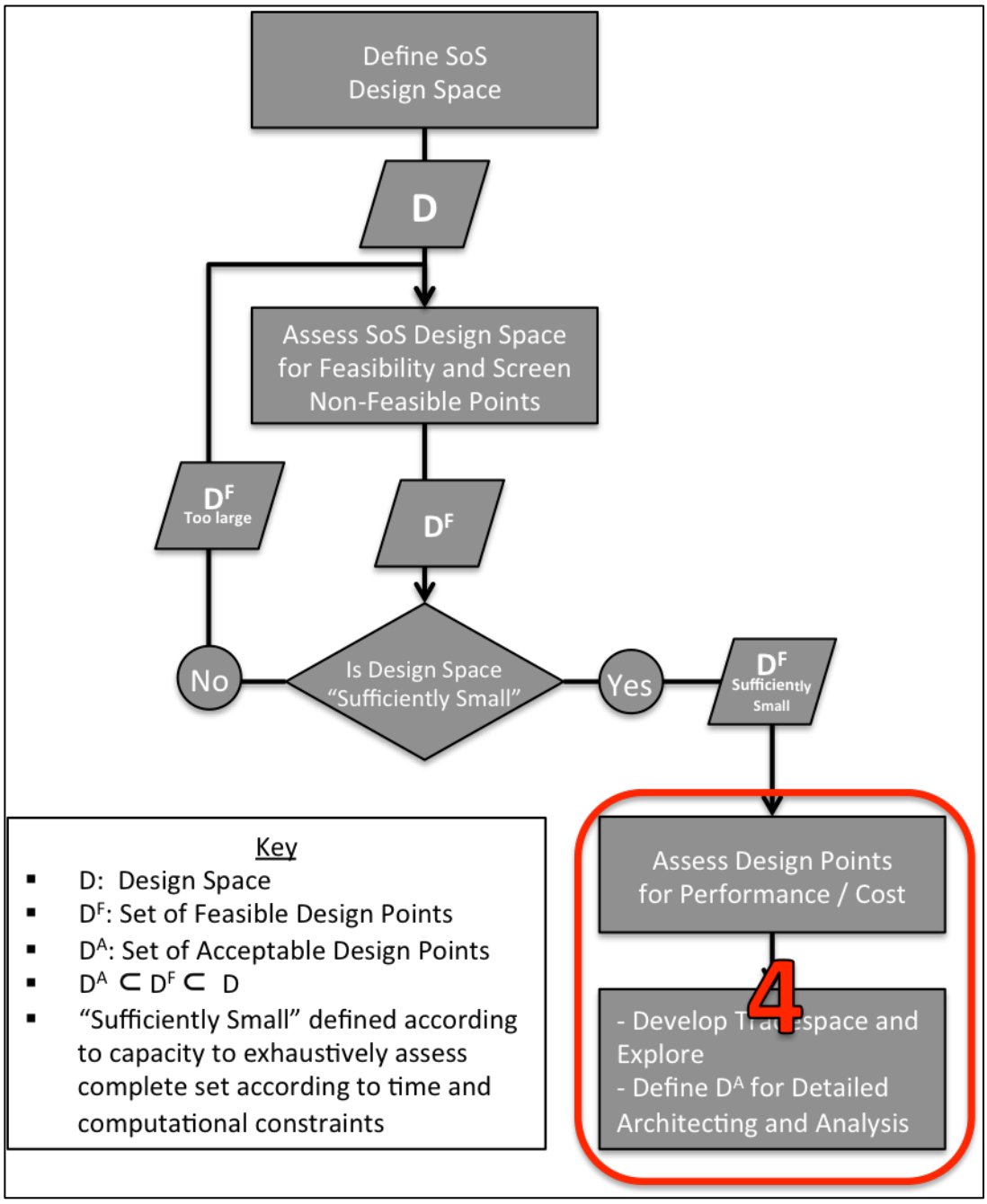


Figure 61. SOS-TDM – Design Point Assessment and Tradespace Analysis

The fourth step of the SoS-TDM is to assess the design points in $\boldsymbol{D^F}$ and use that to develop a tradespace for subsequent analysis. For the IDF SoS, this involved testing

the 7,980 design points judged feasible in two models, an operational ABM and a cost model. This provided the three measures for this scenario, the PTD, PCD, and cost for each design point. The fields of SoS modeling, tradespace exploration, and multi-criteria analysis are well explored (as discussed in Chapter II); however, for completeness, we provide a brief demonstration.

1. IDF-SoS Agent-Based Model

The IDF-SoS ABM assesses a design point's PTD and PCD. It takes a design point as input in the form of a vector: $[S_1, S_2, \ldots S_9, LNO, P_m, ROE, O_o]$ where the variable S_n or LNO is binary, indicating whether or not the n^{th} system is included, P_m is which of four processes is employed, ROE indicates which set of ROE are used, and O_o indicates which of 11 possible organizations are used. It outputs the number of civilian and enemy targets presented and the numbers of each hit; these are used to calculate the PTD and PCD for design point as measured to the nearest percent.

The scenario is a military operation in which targets (enemy and civilian) present themselves in the area of operations (AO) for a pre-determined amount of time. Observers detect, locate, and classify each target according to their P_{detect}, P_{locate}, and $P_{classify}$ respectively. They then use this information and any information (e.g., a request for information from a commander) to choose on which targets to report. Target reports (calls for fire) are then sent to another system in the SoS; the choice of system depends upon the process and organization. Deconflicters, in processes that employ them, aggregate the information they have received to choose targets in accordance with the rules of employment. Deconflicters then send a message to shooters according to the organization. Shooters engage a target location according to the calls for fire they receive or direction from deconflicters; they prioritize the shots fired according to the organization and rules of employment. Throughout this, systems may relay messages that they have received but are not intended for them. Finally, damage is assessed for shots fired and enemy and civilian hits are tallied. For a more detailed treatment of the algorithm, see Appendix B.

The scenario took approximately 1–2 seconds to run each iteration on a personal computer. To test each design point 30 times for a statistically reasonable result took approximately a minute. To test the entire design space took approximately six days of computational time. This was reasonable in the context of the design problem.

2. IDF-SoS Cost Model

The cost model for the IDF-SoS accounted for the cost of each system, the cost for each relationship (to account for training), and the cost for the functions required (again, to account for training). The algorithm to run simply summed the cost of each included system, relationship, and process for a design point (see Appendix B) and output the results. The results are deterministic and only require a single run. The run time was less than a minute for the set of feasible systems.

3. IDF-SoS Tradespace

After running all models, one has defined the tradespace. Recall the mathematical definition of a tradespace from Section II.B.2.g, a design space, its associated attributes and bounding requirements, the six-tuple: $\{\boldsymbol{D}, \boldsymbol{\delta_i}, D_j^{min}, D_j^{max}, \delta_{a*}^{min}, \delta_{a*}^{max}\}$. In this example, the design space, $\boldsymbol{D}$ is defined by the set of vectors of the form:

$$<S_1, S_2, \ldots S_9, LNO, P_m, ROE, O_o>$$

The system attributes, $\delta_{i1}, \delta_{i2}, \delta_{i3}, \text{and } \delta_{i4}$ are feasibility, mean PTD, mean PCD, and cost respectively and defined through the previously described models.[25] The initial bounds are defined as the range of each input parameter and any possible PTD, PCD, or cost. The one requirement that we have already imposed is that $\delta_{*1}^{min} = 1$, i.e., we require a design point to be feasible.

All of the tradespace information is contained in a large spreadsheet with 7,980 rows, corresponding to each design point, and 17 columns, corresponding to the parameters for each design point and the system attributes (performance measures). This,

[25] Note: In the mathematical definition, each system attribute, δ_i is defined by some function. In this case, this function is defined for each input through the model. It is not a "function" in the classic sense such as $y = f(x) = x^2$. It is, however, a function in the sense that it assigns an input to an output.

of course, is not very useful to a decision-maker. Instead, it is presented in a graphical user interface (GUI) as depicted in Figure 62.

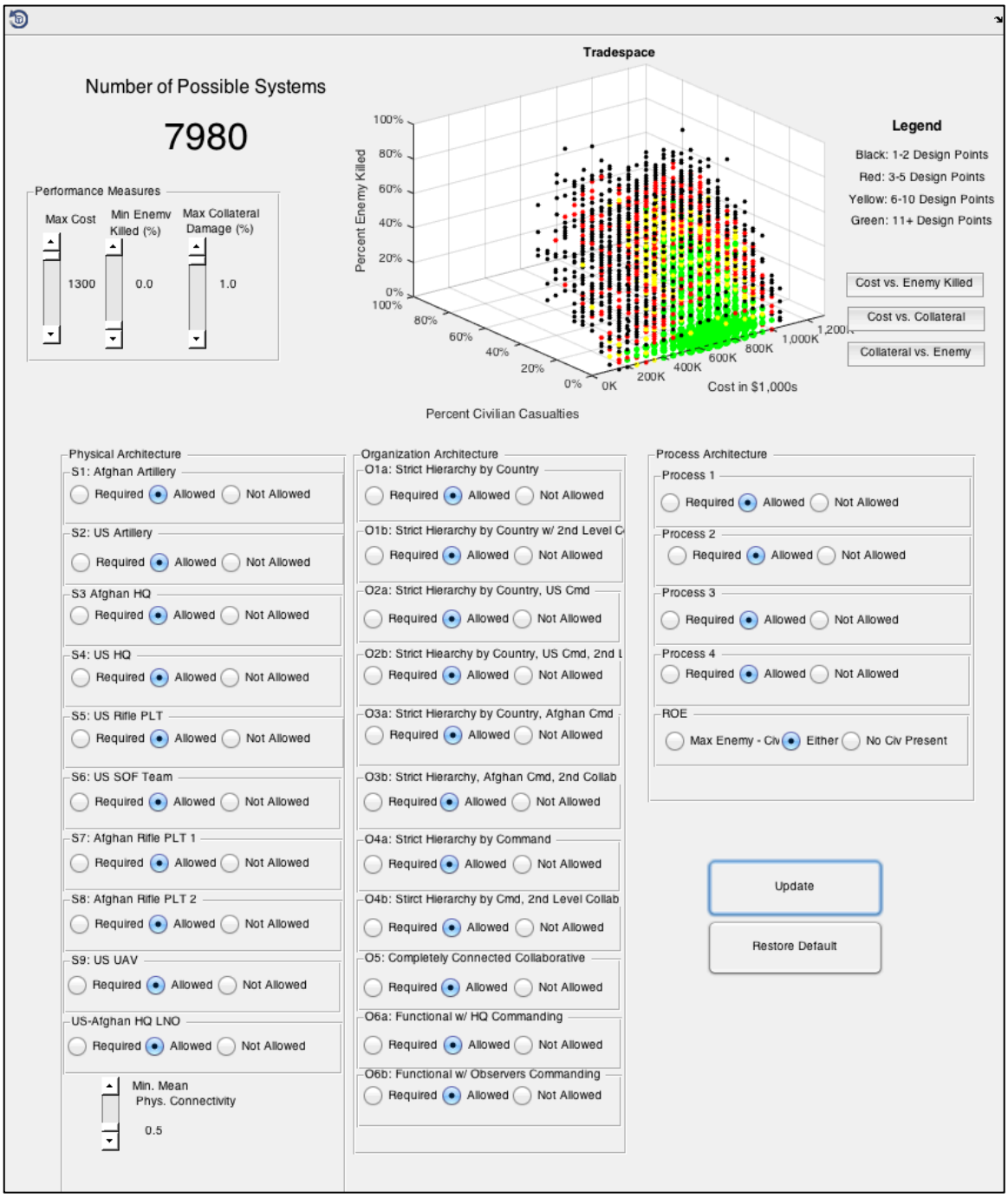


Figure 62. IDF-SoS Tradespace Graphical User Interface (GUI)

The tradespace GUI allows a user to visualize the relationship between the three[26] performance measures as seen in the top right corner of the figure and expanded in Figure 63. Each design point is plotted, to the nearest 5% or $25,000, against its PTD, PCD, and cost. If multiple design points map to the same performance measures, that is indicated in this GUI by varying the color and size of the point as outlined in the figure. One can also choose to view this in two dimensions by selecting the desired option, e.g., collateral damage versus enemy killed as seen in the bottom half of the figure.

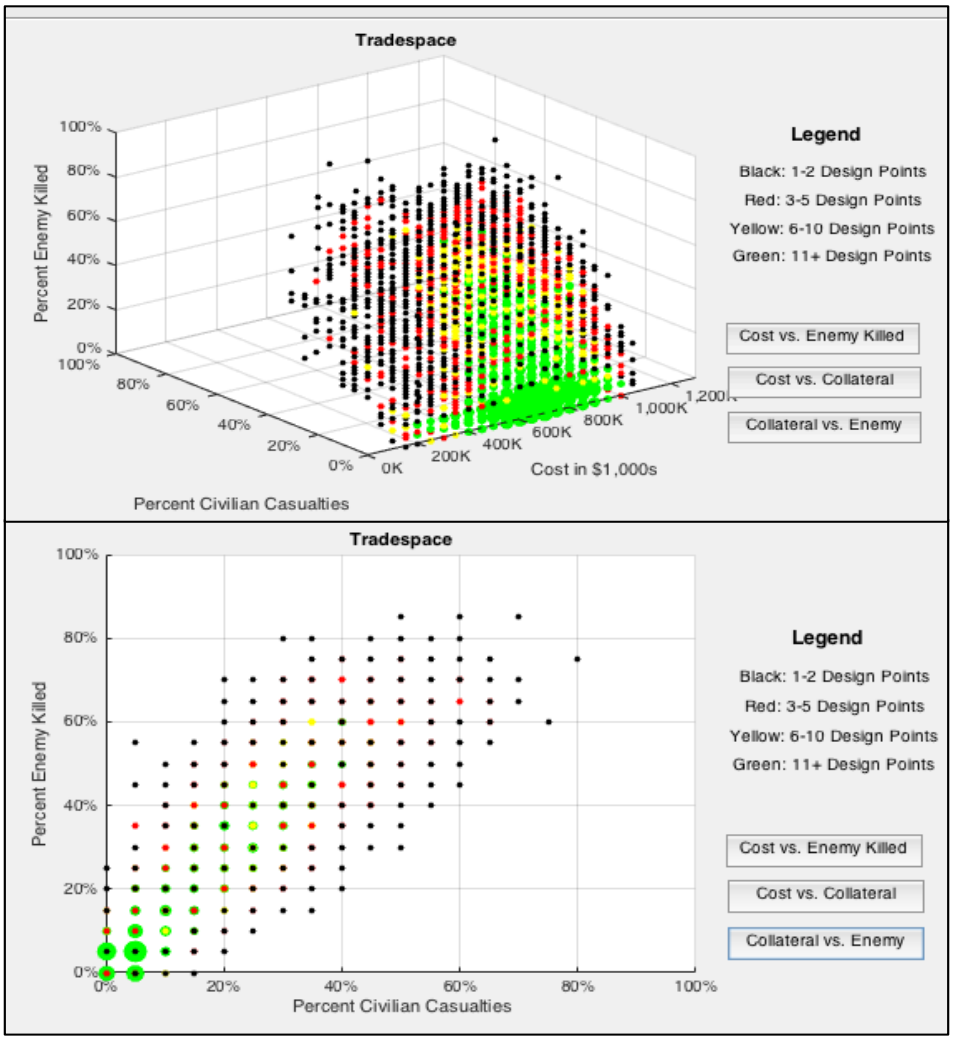
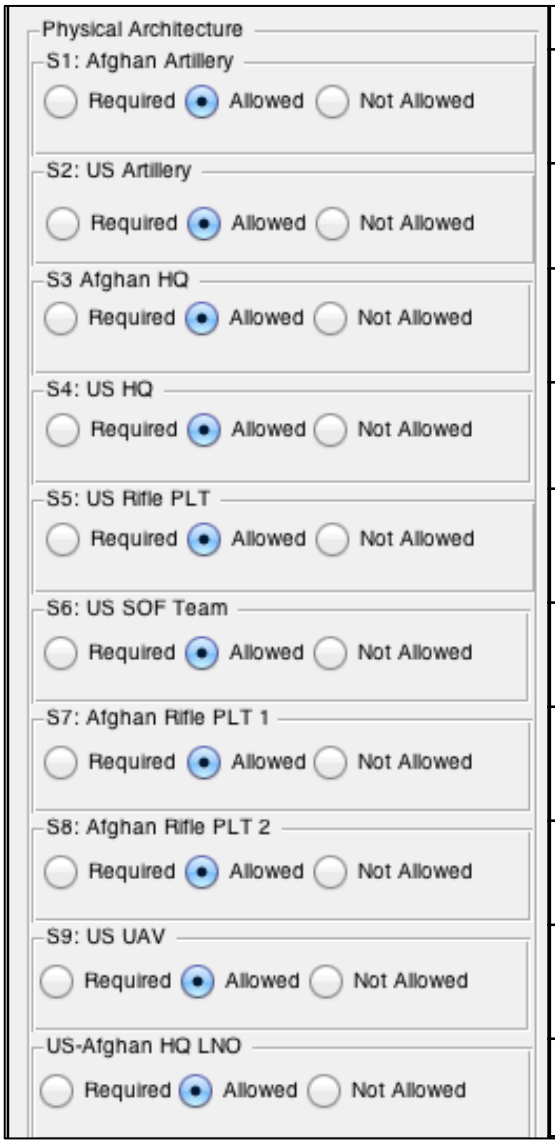


Figure 63. Expanded Projection of Tradespace in Three and Two Dimensions

[26] Direct visualization is possible for two or three performance measures. For four or more, one must select a subset of the performance measures as demonstrated in Beery (2016).

One can vary the bounds of the parameters and attributes, i.e., the $\{D_j^{min}, D_j^{max}, \delta_{a*}^{min}, \delta_{a*}^{max}\}$ to affect the tradespace. This is indicated, for the design parameters, in the three boxes labeled Physical Architecture, Organizational Architecture, and Process Architecture and, for the system attributes, in the box labeled performance measures as seen in Figure 62. As one varies these, the set of acceptable SoS, $\boldsymbol{D^A}$ varies, and the displayed design points changes to only display those that are acceptable.

More explicitly, for the parameters, the terms "Allowed," "Required," and "Not Allowed" vary the bounds of the parameter space, i.e., the D_j. "Allowed" means that an SoS with or without a given system, organization, or process may be included in the set of acceptable SoS ($\boldsymbol{D^A}$). "Not Allowed" means that an SoS with that system, organization, or process may not be included in the set of acceptable SoS ($\boldsymbol{D^A}$). "Required" means only those SoS with the given system, organization, or process are included in the set of acceptable SoS ($\boldsymbol{D^A}$). For example, for the physical architecture, the relationship between the GUI and the mathematical formalization of the design space is demonstrated in Figure 64. There is a direct mapping between the GUI button and a change in the set of design parameters that define $\boldsymbol{D^A}$.



	Required	Allowed	Not Allowed
S1	$D_1 = [1]$	$D_1 = [0, 1]$	$D_1 = [0]$
S2	$D_2 = [1]$	$D_2 = [0, 1]$	$D_2 = [0]$
S3	$D_3 = [1]$	$D_3 = [0, 1]$	$D_3 = [0]$
S4	$D_4 = [1]$	$D_4 = [0, 1]$	$D_4 = [0]$
S5	$D_5 = [1]$	$D_5 = [0, 1]$	$D_5 = [0]$
S6	$D_6 = [1]$	$D_6 = [0, 1]$	$D_6 = [0]$
S7	$D_7 = [1]$	$D_7 = [0, 1]$	$D_7 = [0]$
S8	$D_8 = [1]$	$D_8 = [0, 1]$	$D_8 = [0]$
S9	$D_9 = [1]$	$D_9 = [0, 1]$	$D_9 = [0]$
LNO	$D_{10} = [1]$	$D_{10} = [0, 1]$	$D_{10} = [0]$

Figure 64. Tradespace GUI Design Parameter Bounding to Mathematical Formalization

For the system attribute bounds, as defined by the performance measures, the process is the same. The box labeled "Performance Measures" sets an upper or lower bound on what is acceptable for each performance measure. Formally, this varies the $\{\delta_{a*}^{min}, \delta_{a*}^{max}\}$ associated with each attribute as demonstrated in Figure 65.

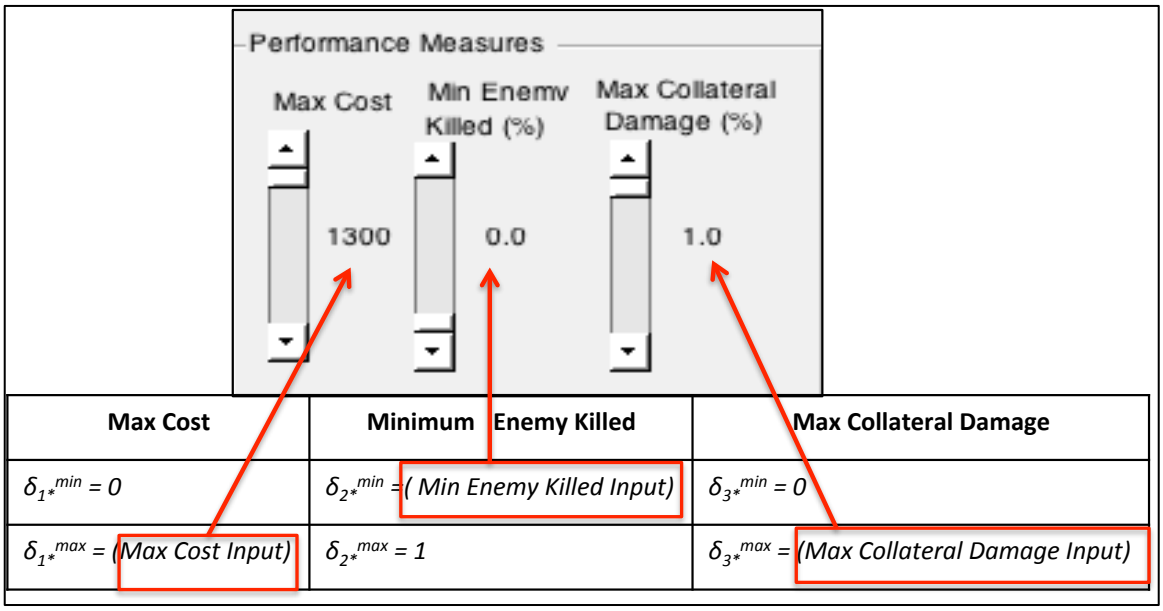


Figure 65. Tradespace GUI System Attribute (Performance Measure) to Mathematical Formalization

With this framework in place, a user, an engineer or decision-maker, may vary the bounds of what is acceptable both in terms of design parameters and performance measures and "explore" the tradespace, i.e., choose a variety of bounding sets that define sets of design points that are feasible. The actual exploration of a tradespace has been reviewed in the literature (e.g., Ross and Hastings 2005; PSU-ARL 2015; Beery 2016) and outside the scope of this research; a more detailed treatment of tradespace exploration for this example is seen in Appendix B.

F. CONCLUSION

In this example, we use the SoS-TDM to develop the tradespace of an IDF-SoS. Through the use of the SoS-TDM and SoS-AFAM we winnowed the design space from 90,112 points to 7,980 (9%) feasible design points. This allowed us to exhaustively assess the set of feasible design points for operational performance through the use of an ABM in less than a week of computational time using a personal computer. This would have been infeasible for the entire design space, as it would have taken 1,500 hours (two months) of computing time to assess all 90,112 points. Furthermore, assessing the 82,132

infeasible design points would have been wasted effort as those points could not be realized, even if they produced acceptable performance measures.

To winnow the initial design space of 90,112 points took less than less than ten minutes of computational time. By partitioning the design space, we only had to assess:

1. C = 1,024 physical design points that resulted in 372 physically feasible points. Thus $x = \frac{372}{1{,}024} = 0.36$.
2. xCP = 372×8 = 2,976 process design points that resulted in 1,938 process feasible design points. Thus $y = \frac{1{,}938}{2{,}976} = 0.65$.
3. xCO = 372×11 = 4,092 organizational design points that resulted in 1,677 organizationally feasible design points. Thus $z = \frac{1{,}677}{4{,}092} = 0.41$.
4. $wxCOP$ = 1,677×8 = 13,416 complete design points.

In total, we made 1,024+2,976+4,092+13,416 = 21,508 feasibility assessments. Thus, $\Pi = \frac{21{,}508}{90{,}112} = 0.24$. This analysis resulted in 7,980 feasible SoS design points.

The end result of using the SoS-TDM and SoS-AFAM for the IDF SoS was a tradespace that defined an SoS across its physical, organizational, and process parameters and against three performance measures. The inclusion of all three SoS architectural perspectives allowed better fidelity SoS modeling and simulation as it organization and process parameters are key to ABM and SoS analysis. The resultant tradespace GUI is a user friendly method of exploratory design decision making that may be used to define a small subset of designs for subsequent detailed architecting and analysis. Importantly, each design point includes the necessary design parameters for complete SoS architecting.

V. CONCLUSION

A. SUMMARY

Contemporary organizations desire to explicitly engineer SoS; however, this has proven difficult. A significant aspect of this challenge is that SoS are highly complex—not only are the constituent systems of an SoS managerially and operationally independent, but there are significant interactions among the physical composition, processes used, and organizational relationships of the SoS. A SoS architecture must describe these different perspectives. Moreover, it is through the interactions of these different perspectives that an SoS generates its emergent capabilities. This makes it difficult to easily understand and predict the implications of choosing any set of design parameters.

Within engineering design, there are three methods of design-decision making: heuristic, normative, and exploratory. For SoS, there are heuristic and normative design decision-making methodologies; however, there are limited exploratory SoS design decision-making methodologies, and the ones that do exist make significant simplifying assumptions abstracting away the necessary architectural perspectives of an SoS.

Within the field of MBSE, there has been much effort on developing exploratory design decision-making methods, primarily in the area of tradespace exploration. These methods require one to define the relationship between a design point and its attributes (e.g., cost, performance). This is challenging for an SoS due to the complex nature of the interactions. SoS design points are best assessed individually; however, this limits the number of design points that may be assessed in total due to time and computational constraints.

Taken together, these two challenges—the requirement to design and represent SoS with the considerations of physical, process, and organization and the lack of any current ability to develop the tradespace for an SoS represented in this manner—create a potential for an extension to the state-of-the-art of systems engineering.

To address this challenge, the dissertation developed the SoS Tradespace Definition Methodology and the SoS Architectural Feasibility Assessment Model; both are depicted in Figure 66. The SoS-TDM is a four step methodology in which an engineer 1) defines an SoS design space according to the physical, process, and organization perspectives, 2) assesses these design points for feasibility and winnows the infeasible points, 3) iterates this process until the remaining feasible set is "sufficiently small" for exhaustive analysis, and 4) exhaustively analyzes the set of feasible SoS design points to create a complete tradespace. The SoS-AFAM is how one conducts Step 2 of the SoS-TDM. It involves testing different subsets of the SoS design space for feasibility from a variety of perspectives and then using the results of these tests to define a sub-set of the design space that is feasible.

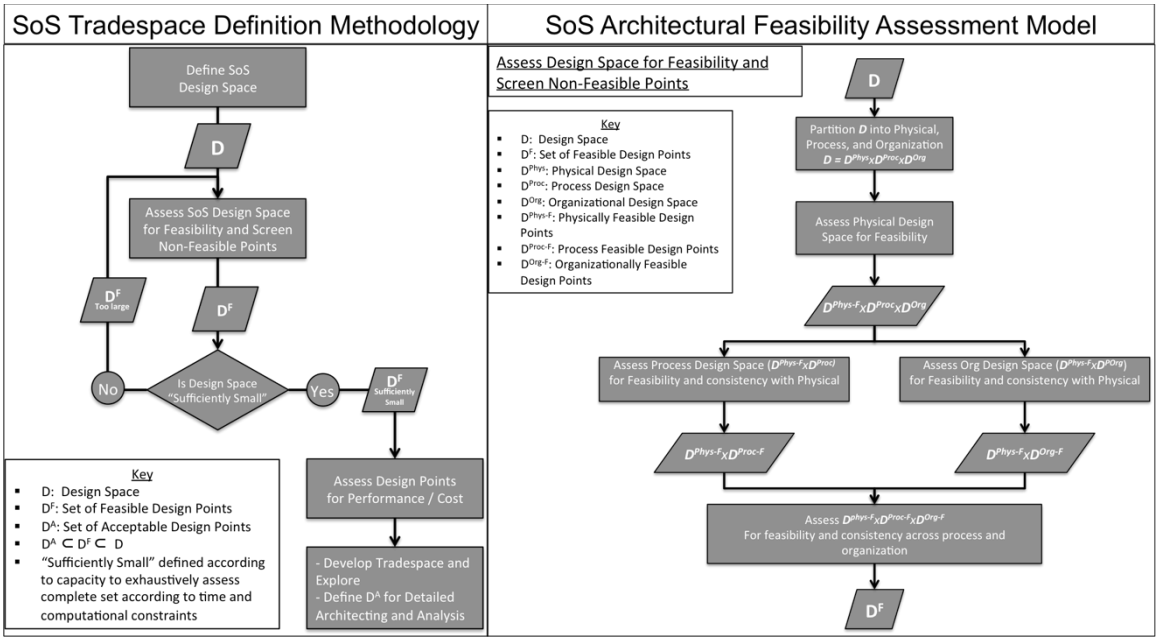


Figure 66. The SoS-TDM and SoS-AFAM

B. CONCLUSIONS

The first chapter identified three research questions to answer to extend the current state-of-the-art.

1. How may the required SoS architectural perspectives of physical, process, and organizational be used to define an SoS design space?

This question is answered Step 1 of the SoS-TDM. Generally speaking, there exist defined architectural representations (e.g. DODAF views) for each of these perspectives. The distinction for the SoS-TDM is that it defines parameters that can define a design space such that these parameters may be used to build the required architectural perspectives.

2. How may one assess the feasibility of an SoS architecture?

The SoS-AFAM presents a model to define the feasibility of an SoS design point. This involves a series of logical tests that assess different aspects of the design space. These tests include requirements for the physical (communications) topology and organizational topology to form connected networks, for the included systems to provide sufficient functionality to complete the desired processes, system acceptance of the necessary organizational relationships and rules of employment, and other more in depth considerations that refine these basic questions. These tests must be answered positively for any SoS to be realized as, if they are not, there exists a system in the SoS that either does not agree to the requirements placed upon it or is not connected to the other systems in the SoS.

3. May the above be used to define an SoS tradespace in an efficient manner so that it can be incorporated into existing MBSE TSE methodologies?

The SoS-TDM is a method to define and winnow, via the SoS-AFAM, an SoS tradespace in a reasonable time. It does this by introducing feasibility tests to selectively choose a significantly smaller subset of the SoS design space for analysis. While it is impossible to prove that the feasible subset of the design space will always be sufficiently small, it is often the case as the complexity of an SoS makes the likelihood of all feasibility requirements being met fairly low.

In general, one may assess an SoS design space for feasibility fairly quickly with the SoS-AFAM. This is done by partitioning the design space and assessing these partitions prior to assessing the total design space. The percent of the design space one must assess is:

$$\Pi = \frac{1}{OP} + \frac{x}{P} + \frac{x}{O} + wx$$

where O is the number organizational points, P is the number of process points, x is the percentage of physical compositions that are feasible and w is the minimum of the percentage of organizational or process designs that are feasible. Each feasibility tests has varying algorithmic complexity as described in Table 11. In the example of the IDF-SoS, the computational time to assess feasibility for all 90,112 points was less than ten minutes.

Finally, the SoS-TDM extends current state-of-the-art MBSE methodologies, notably the MEASA as depicted in Figure 67. As the SoS-TDM starts and ends at the same point as the MBSE MEASA, one may integrate it into greater MBSE methodologies as described by Beery (2016).

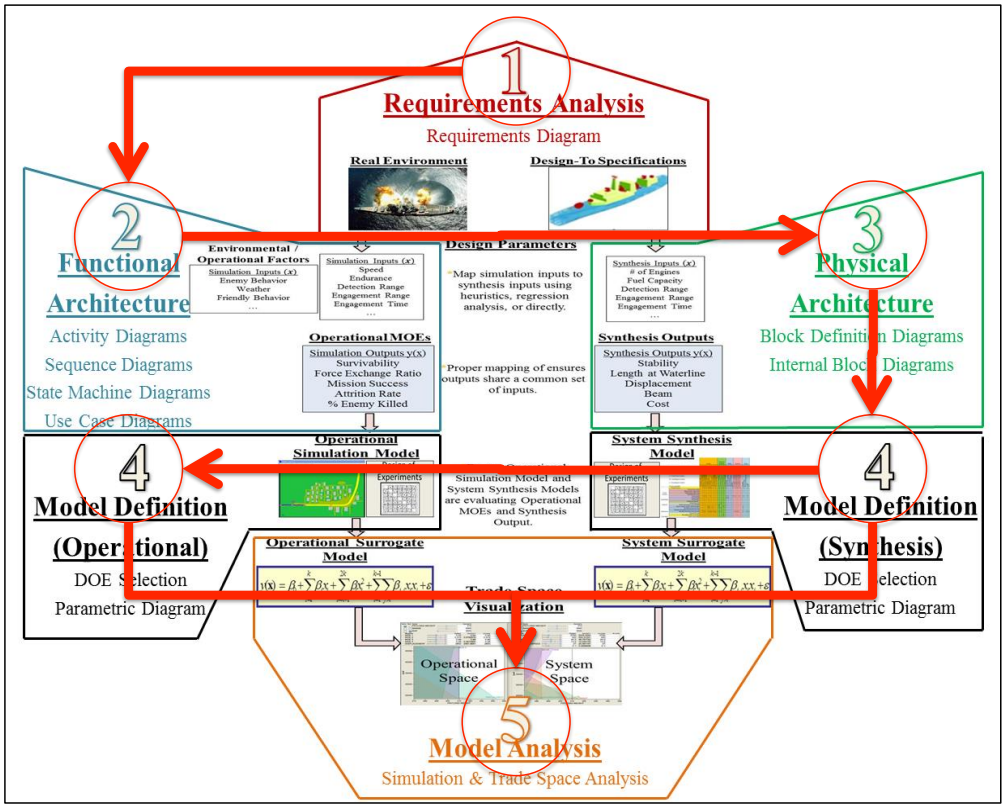


Figure 67. SoS-TDM Modification of the MBSE MEASA. Adapted from Beery (2016)

C. FUTURE RESEARCH

There are at least seven areas of potential future research to extend and improve this methodology and model. These include detailed architecting and analysis, process and organization definition, collaborative SoS, transfer functions, strategic SoS design decision-making, variable environments, and unanticipated emergent behaviors.

The first area of further research is in applying the SoS-AFAM to greater levels of detailed architecting and analysis. As presented, the feasibility of a system is generally binary—an SoS design is feasible or it is not, it is connected or not, the physical supports the organization or it does not. In some cases, there are gradations of feasibility as feasibility itself is defined by a decision-maker's requirements (e.g., a system that may be realized in one year may or may not be feasible depending upon the decision-maker's timeline). Furthermore, the analysis made the simplifying assumption that the information used was fungible and could be passed across any network with only some translation time to switch between different networks; this may not generally be true. Moreover, there may be multiple, different types of information (or possibly other resources) that must transition between systems. In some cases, certain types of information or other resources may only require a transition between a small subset of an SoS and not the whole SoS. For example, one may have a supply chain SoS that must be completely connected by information sharing, but only requires a sub-set of it that must form a connected physical network of actual material exchange.

The second area of continued research is with regard to how one defines processes and organizations. As presented, an engineer uses heuristics to define multiple distinct processes and organizations. This solves a combinatorial problem as there are essentially infinite ways in which one could arrange even a small number of functions to define an operational activity flow, rules one could come up with, or organizations one could define with even a small number of operational activities or organizational relationships. A more analytic tool to define and assess potential processes and organizations may further extend Step 1 of the SoS-TDM—how we define the SoS design space. For example, one may define a set of available operational activities and relationships and assess which sets present a desired emergent behavior and then assess

what set of systems could support those designs. This may be of particular use in a concept related to SoS—Families of Systems or swarms (groups of modular, but distinct systems that interact to provide a desired capability).

A third area of research involves extending the scope of the SoS-TDM and SoS-AFAM to collaborative SoS. As developed, the SoS-TDM and SoS-AFAM only apply to acknowledged and directed systems. This makes the tacit assumption that, if an SoS is found to be feasible (and, in particular organizationally acceptable), one can develop that SoS. For collaborative SoS, however, one must place greater emphasis on the incentive structure as a function of the architecture. As a part of considering which systems to incentivize the most, one may consider which systems are most important to providing a capability or performance measure. Game theory suggests different ways in which one can consider which member of a team (i.e., constituent system in a system) should be rewarded based upon how the team performs (i.e., its performance measures) with or without that member (e.g., the Shapley Value).[27]

A fourth area of future research is in regard to transfer functions as defined in Section II.C.1.b. Recall that a transfer function is a function that takes a set of design parameters as input and outputs an operational parameter for use in an operational simulation. They are useful for the practical purpose that operational and system synthesis models often require different inputs. A simple, if obvious example of an SoS transfer function would be to define the latency and accuracy of a message passed between two systems over its organization and physical architectures. This could be used in a non-ABM simulation to assess SoS operational performance.

A fifth area is that the SoS-TDM and SoS-AFAM, as developed, only consider a single design phase in an SoS's life-cycle. In reality, the development of an SoS is an iterative process that occurs repeatedly as constituent systems change over their own life-cycles. As presented, the SoS-TDM and SoS-AFAM only consider the impacts of an SoS design for this point in time. This is a tactical perspective. A more strategic perspective would consider the impact of an SoS design over a longer life-cycle, particularly as it

[27] The Shapley Value is, in essence, a measure of how much a given player contributes relative to the other players in a cooperative game (Owen 2013).

relates to the life-cycle of its constituent systems and their potential replacements. Ideally, one would want a methodology that introduced a method of measuring an SoS's performance over multiple iterations of its life-cycle and its ability to continue to provide utility to its stakeholders.

A sixth area is that, as mentioned in Chapter II, this dissertation assumed a static environment. Systems must operate in many environments. A changing environment varies system attributes (e.g., a system may perform well in one environment and not well in another). This, in turn, varies the tradespace, both in how one must define it and how one explores it. Moreover, a varying environment may affect SoS feasibility. For example, two systems may share a FM radio communications sub-system. The range of the FM radio varies depending upon the terrain. Thus, an SoS that depends upon this connection may or may not be feasible depending upon the terrain. Similarly, relationships between two systems may vary in acceptability depending upon the political environment.

Finally, the SoS-TDM and SoS-AFAM are focused on the trades among pre-defined performance measures, in other words, among expected emergent properties. A challenge of SoS engineering is that there are often unexpected emergent properties (Keating 2009). Understanding the nature of these emergent properties—what combination of systems and interactions lead to them—is highly useful. Conceptually, if an unexpected emergent property is one that may be modeled (i.e., it is simple or weak per Maier's (2015) definition), it can potentially be observed in a simulation of that SoS. If one has a method to identify unexpected emergent properties that occur in the modeling and simulation of the set of all feasible SoS identified by the SoS-AFAM, one could use that information to identify which combinations of systems and their interactions cause that unexpected emergent property. Accordingly, developing a method for identifying unexpected emergent properties in a simulation would be highly useful future research.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. DEPARTMENT OF DEFENSE ARCHITECTURE FRAMEWORK

The Department of Defense Architecture Framework (DODAF) is the DOD's required architecture framework for its systems. It is referenced throughout this dissertation. The most recent version is DODAF 2.02 (DOD CIO 2010).

Importantly, the DODAF is focused upon defining the necessary data that describes the system and may subsequently be turned into models that demonstrate a particular view (DOD CIO 2010). This data is called the DODAF Metal Model (DM2) (DOD CIO 2010). This is necessary for actually building an architecture in accordance with DODAF; however, of greater conceptual interest are the various views that use this data.

DODAF does not prescribe the actual depiction of any particular viewpoint (DODAF CIO 2010); however the Object Management Group (OMG) has developed a set of standards for the Unified Modeling Language (UML) for DODAF called the Unified Profile for DODAF / MODAF (UPDM) (Object Management Group [OMG] 2016). The company No Magic provides a quick reference guide that provides examples of employment of the UPDM available on their website, http://www.nomagic.com/support/quick-reference-guides.html.

A. ALL VIEWPOINT (AV)

The All Viewpoint (AV) provides an overview of the architecture and defines constraints, requirements, objectives, and key terms for the architecting (as opposed to the system being architected). It is composed of the AV-1: Overview and Summary Information and AV-2: Integrated Dictionary (DOD CIO 2010).

B. CAPABILITY VIEWPOINT (CV)

The Capability Viewpoint (CV) describes the capability of the system under development and its relationship to other systems (DOD CIO 2010). It is composed of seven models: CV-1: Vision, CV-2: Capability Taxonomy, CV-3: Capability Phasing,

CV-4 Capability Dependencies, CV-5: Capability to Organizational Development Mapping, CV-6: Capability to Operational Activities Mapping, CV-7: Capability to Services Mapping. Each is described in Figure 68.

Model	Description
CV-1: Vision	Addresses the enterprise concerns associated with the overall vision for transformational endeavors and thus defines the strategic context for a group of capabilities.
CV-2: Capability Taxonomy	Captures capability taxonomies. The model presents a hierarchy of capabilities. These capabilities may be presented in context of a timeline - i.e., it can show the required capabilities for current and future capabilities.
CV-3: Capability Phasing	The planned achievement of capability at different points in time or during specific periods of time. The CV-3 shows the capability phasing in terms of the activities, conditions, desired effects, rules complied with, resource consumption and production, and measures, without regard to the performer and location solutions
CV-4: Capability Dependencies	The dependencies between planned capabilities and the definition of logical groupings of capabilities.
CV-5: Capability to Organizational Development Mapping	The fulfillment of capability requirements shows the planned capability deployment and interconnection for a particular Capability Phase. The CV-5 shows the planned solution for the phase in terms of performers and locations and their associated concepts.
CV-6: Capability to Operational Activities Mapping	A mapping between the capabilities required and the operational activities that those capabilities support.
CV-7: Capability to Services Mapping	A mapping between the capabilities and the services that these capabilities enable.

Figure 68. DODAF Capability Viewpoints. Source: DOD CIO (2010)

C. DATA AND INFORMATION VIEWPOINT (DIV)

The Data and Information Viewpoint (DIV) describes information requirements and rules from Conceptual (DIV-1), Logical (DIV-2), and Physical (DIV-3) perspectives. Each is described in Figure 69.

DIV-1: Conceptual Data Model	The required high-level data concepts and their relationships.
DIV-2: Logical Data Model	The documentation of the data requirements and structural business process (activity) rules. In DoDAF V1.5, this was the OV-7.
DIV-3: Physical Data Model	The physical implementation format of the Logical Data Model entities, e.g., message formats, file structures, physical schema. In DoDAF V1.5, this was the SV-11.

Figure 69. DODAF Data and Information Viewpoints. Source: DOD CIO (2010)

D. OPERATIONAL VIEWPOINT

The DODAF describes nine (broken into six types, with several sub-types) Operational Viewpoints that "describe the tasks and activities, operational elements, and resource flow exchanges required to conduct operations" (DOD CIO 2010). These facilitate understanding of how the system is employed and operates, its goals, and how it interacts with its environment. These viewpoints are described in Figure 70.

Model	Description
OV-1: High-Level Operational Concept Graphic	The high-level graphical/textual description of the operational concept.
OV-2: Operational Resource Flow Description	A description of the Resource Flows exchanged between operational activities.
OV-3: Operational Resource Flow Matrix	A description of the resources exchanged and the relevant attributes of the exchanges.
OV-4: Organizational Relationships Chart	The organizational context, role or other relationships among organizations.
OV-5a: Operational Activity Decomposition Tree	The capabilities and activities (operational activities) organized in a hierarchal structure.
OV-5b: Operational Activity Model	The context of capabilities and activities (operational activities) and their relationships among activities, inputs, and outputs; Additional data can show cost, performers or other pertinent information.
OV-6a: Operational Rules Model	One of three models used to describe activity (operational activity). It identifies business rules that constrain operations.
OV-6b: State Transition Description	One of three models used to describe operational activity (activity). It identifies business process (activity) responses to events (usually, very short activities).
OV-6c: Event-Trace Description	One of three models used to describe activity (operational activity). It traces actions in a scenario or sequence of events.

Figure 70. DODAF Operational Viewpoints. Source DOD CIO (2010)

E. PROJECT VIEWPOINT (PV)

The DODAF Project Viewpoint (PV) describes the information necessary for the various program management activities required to bring a system into being (DOD CIO 2010). These are described in Figure 71.

PV-1: Project Portfolio Relationships	It describes the dependency relationships between the organizations and projects and the organizational structures needed to manage a portfolio of projects.
PV-2: Project Timelines	A timeline perspective on programs or projects, with the key milestones and interdependencies.
PV-3: Project to Capability Mapping	A mapping of programs and projects to capabilities to show how the specific projects and program elements help to achieve a capability.

Figure 71. DODAF Project View Points. Source DOD CIO (2010)

F. SERVICES VIEWPOINT (SVCV)

The DODAF Service Viewpoint (SvcV) describes the services and their interconnections provided to or from the system being modeled (DOD CIO 2010). These are described in Figure 72.

SvcV-1 Services Context Description	The identification of services, service items, and their interconnections.
SvcV-2 Services Resource Flow Description	A description of Resource Flows exchanged between services.
SvcV-3a Systems-Services Matrix	The relationships among or between systems and services in a given Architectural Description.
SvcV-3b Services-Services Matrix	The relationships among services in a given Architectural Description. It can be designed to show relationships of interest, (e.g., service-type interfaces, planned vs. existing interfaces).
SvcV-4 Services Functionality Description	The functions performed by services and the service data flows among service functions (activities).
SvcV-5 Operational Activity to Services Traceability Matrix	A mapping of services (activities) back to operational activities (activities).
SvcV-6 Services Resource Flow Matrix	It provides details of service Resource Flow elements being exchanged between services and the attributes of that exchange.
SvcV-7 Services Measures Matrix	The measures (metrics) of Services Model elements for the appropriate timeframe(s).
SvcV-8 Services Evolution Description	The planned incremental steps toward migrating a suite of services to a more efficient suite or toward evolving current services to a future implementation.
SvcV-9 Services Technology & Skills Forecast	The emerging technologies, software/hardware products, and skills that are expected to be available in a given set of time frames and that will affect future service development.
SvcV-10a Services Rules Model	One of three models used to describe service functionality. It identifies constraints that are imposed on systems functionality due to some aspect of system design or implementation.
SvcV-10b Services State Transition Description	One of three models used to describe service functionality. It identifies responses of services to events.
SvcV-10c Services Event-Trace Description	One of three models used to describe service functionality. It identifies service-specific refinements of critical sequences of events described in the Operational Viewpoint.

Figure 72. DODAF Services Viewpoints. Source: DOD CIO (2010)

G. STANDARDS VIEWPOINT (STDV)

The DODAF Standards Viewpoint (StdV) describes the various internal interactions and interdependencies of the system (DOD CIO 2010). They are described in Figure 73.

StdV-1 Standards Profile	The listing of standards that apply to solution elements.
StdV-2 Standards Forecast	The description of emerging standards and potential impact on current solution elements, within a set of time frames.

Figure 73. DODAF Standards Viewpoints. Source: DOD CIO (2010).

H. SYSTEMS VIEWPOINT (SV)

The DODAF Systems Viewpoint (SV) describes the "systems and interconnections providing for, or supporting, DOD functions" (DOD CIO 2010). This is a particularly useful view for the SoS-TDM and SoS-AFAM when developing an SoS composed of DOD systems. The thirteen views are described in Figure 74.

Models	Descriptions
SV-1 Systems Interface Description	The identification of systems, system items, and their interconnections.
SV-2 Systems Resource Flow Description	A description of Resource Flows exchanged between systems.
SV-3 Systems-Systems Matrix	The relationships among systems in a given Architectural Description. It can be designed to show relationships of interest, (e.g., system-type interfaces, planned vs. existing interfaces).
SV-4 Systems Functionality Description	The functions (activities) performed by systems and the system data flows among system functions (activities).
SV-5a Operational Activity to Systems Function Traceability Matrix	A mapping of system functions (activities) back to operational activities (activities).
SV-5b Operational Activity to Systems Traceability Matrix	A mapping of systems back to capabilities or operational activities (activities).
SV-6 Systems Resource Flow Matrix	Provides details of system resource flow elements being exchanged between systems and the attributes of that exchange.
SV-7 Systems Measures Matrix	The measures (metrics) of Systems Model elements for the appropriate timeframe(s).
SV-8 Systems Evolution Description	The planned incremental steps toward migrating a suite of systems to a more efficient suite, or toward evolving a current system to a future implementation.
SV-9 Systems Technology & Skills Forecast	The emerging technologies, software/hardware products, and skills that are expected to be available in a given set of time frames and that will affect future system development.
SV-10a Systems Rules Model	One of three models used to describe system functionality. It identifies constraints that are imposed on systems functionality due to some aspect of system design or implementation.
SV-10b Systems State Transition Description	One of three models used to describe system functionality. It identifies responses of systems to events.
SV-10c Systems Event-Trace Description	One of three models used to describe system functionality. It identifies system-specific refinements of critical sequences of events described in the Operational Viewpoint.

Figure 74. DODAF Systems Viewpoints. Source: DOD CIO (2010)

APPENDIX B. ADDITIONAL INFORMATION FROM THE IDF-SOS

The actual modeling and simulation of SoS is a well-defined and well-understood field of study outside the scope of this research. Furthermore, the actual exploration of an SoS tradespace is conducted in a manner similar to any other multi-dimensional tradespace exploration and is also outside the scope of this research. This appendix provides additional details regarding the IDF SoS model and simulation that were necessary for the demonstration, but extraneous from the main purpose of the research.

A. CONSTITUENT SYSTEM INFORMATION

1. Shooters

The first set of systems are those that provide the function "shoot." To shoot is to propel a projectile from one location to another. The shooting function is measured by two MOEs, "Probability of a Hit" and "Probability of a Kill." Probability of a Hit (P_{hit}) is the probability that a system will hit the location at which it aimed. Note that once a round is fired, it will land somewhere. There is a $1 - P_{hit}$ probability that the fired rounds land at a location other than the one the shooter aimed at. Probability of a Kill (P_{kill}) is the probability that a shot fired will kill a target at the location where the round impacts. This is assessed independently for each target at that location. Note that these measures are high-level generalizations of other performance measures that may be considered in higher fidelity models such as: rounds fired per target, rounds per minute, time of flight, or explosive radius. Finally, the shooting systems also have a "memory" which assesses how long a system can remember the information (regarding targets) passed to it.

a. System 1 – Afghan Army Artillery Battery

The first available system is an Afghan Army Artillery Battery. This a unit of four to six artillery pieces, such as the 122mm D-30 howitzer, an artillery piece originally made in the former Soviet Union. The Afghan Army has historically lacked advanced training, communications, and equipment; thus, this system is less accurate than a comparable American one.

b. System 2 – U.S. Army Artillery Battery

A U.S. Army artillery battery is composed of six howitzers. For this example, the howitzers are M198 or M777 (the updated version of the M198) 155mm howitzers. U.S. Army artillery units have modern equipment, communications, and significant training that results in highly accurate fires.

2. Deconflicters

The second important function is aggregating and deconflicting the information presented in the SoS to maximize the potential for enemy killed and minimize the potential for civilian casualties. These systems, the deconflicters, do this through collecting information, developing it into a world view, and then using this world view to make decisions. The primary metric by which the deconflicters may be measured is through their "memory," that is, how much information can they store and process.

a. System 3 – Afghan Army Kandak (Battalion) Headquarters

An Afghan Army Kandak (the equivalent of a U.S. battalion) is typically commanded by a lieutenant colonel and has a staff that facilitates information processing and decision-making, but limited communications capabilities. Afghan units, such as the aforementioned artillery battery and to be described rifle platoons, habitually report to the Kandak headquarters.

b. System 4 – U.S. Army Battalion Headquarters

A U.S. Army battalion headquarters, also commanded by a lieutenant colonel, has a robust staff and communications equipment that are capable of receiving and processing significant amounts of information and directing the activities of subordinate and collaborating units.

3. Observers

a. System 5 – U.S. Army Rifle Platoon

A U.S. Army Rifle Platoon is an infantry unit of approximately 40 soldiers. Importantly, for this example, the platoon has a fire support team of a forward observer

and his radio-telephone operator (RTO) who specialize in identifying targets and calling for indirect fire. They typically have significant training in this area, and equipment for observing, locating, and identifying targets. In this situation, similar to recent modern experiences, it is difficult, however, for U.S. observers to distinguish between civilian targets and enemy targets masquerading as civilians.

b. System 6 – U.S. Special Operations Forces Team

A U.S. SOF Team is a 12-soldier team trained in various specialties. In particular, there are soldiers trained and equipped for indirect fire observation to the same or superior levels as the forward observer team in a Rifle Platoon. Moreover, SOF Teams are trained and equipped to work with and communicate with foreign military forces, thus enabling them to communicate with the Afghan forces in this example.

Although a Special Operations team is *de jure* a member of the U.S. Army, Air Force, Navy, or Marines, special operations have evolved to such an extent that the command and its subordinate systems are, *de facto,* independent of their separate services.

c. System 7 and System 8 – Afghan Army Rifle Platoons 1 and 2

For this example, there are two identical systems, Systems 7 and 8. Both are Afghan Army Rifle Platoons. These are similar to U.S. Army Rifle Platoons, but they lack the level of training and equipment, resulting in less accurate calls for fire; however, being local forces, they are more likely to correctly distinguish between civilian and enemy targets.

d. System 9 – U.S. Air Force Unmanned Aerial Vehicle

The final potential system is a U.S. Air Force Unmanned Aerial Vehicle (UAV). It provides full motion video observation. It can only observe a small section of the area of operations at any time, and thus has a relatively low likelihood of identifying a target; however, if it does identify the target, it very capable at providing an accurate location.

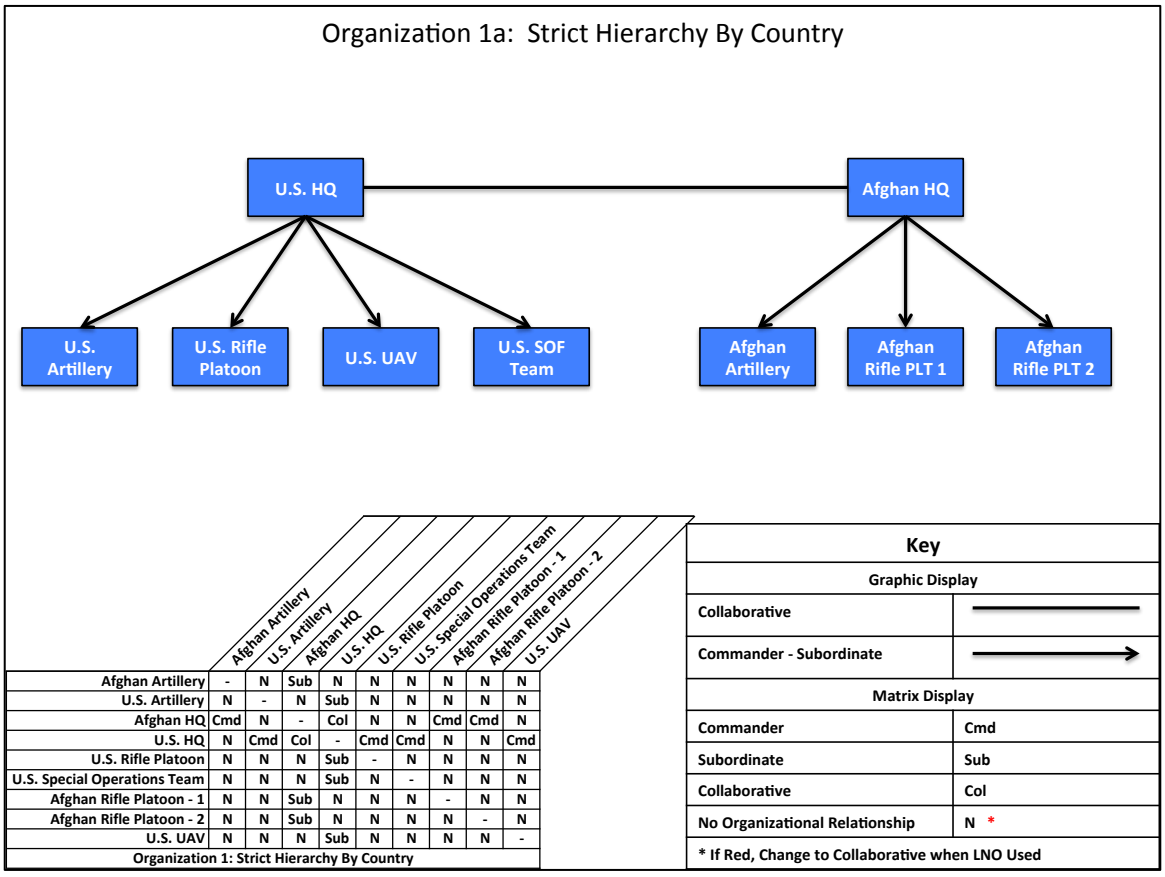
4. Communication Systems

The potential communication systems for this SoS are the communication sub-systems each system has. Furthermore, there is the potential to refactor one system, the "U.S. Headquarters," so that it can communicate on the "Afghan FM" (by adding an "Afghan Liaison"). The five communications systems are:

- Afghan FM Radio: This is a standard two-way FM radio. The language spoken on this channel is Dari (an Afghan language). The radio itself is unencrypted.
- One Station Remote Video Terminal (OSRVT): This is a U.S. military system that allows a UAV to provide video feed to the user and for the user to communicate with the UAV.
- U.S. FM Radio: This is a standard two-way FM radio. The language spoken is English and conforms to all normal military radio standards. The radio transmissions are encrypted.
- Blue Force Tracker (BFT): This is a U.S. military system in which users can see each other's location on a map (based upon their GPS signal) and send text messages.
- My Internet Relay Chat (MIRC): Is an encrypted computer chat program used by the U.S. military. It functions like any other sort of Internet instant messenger chat.

B. ORGANIZATION DEPICTIONS

Each full size organization description is depicted in this section.

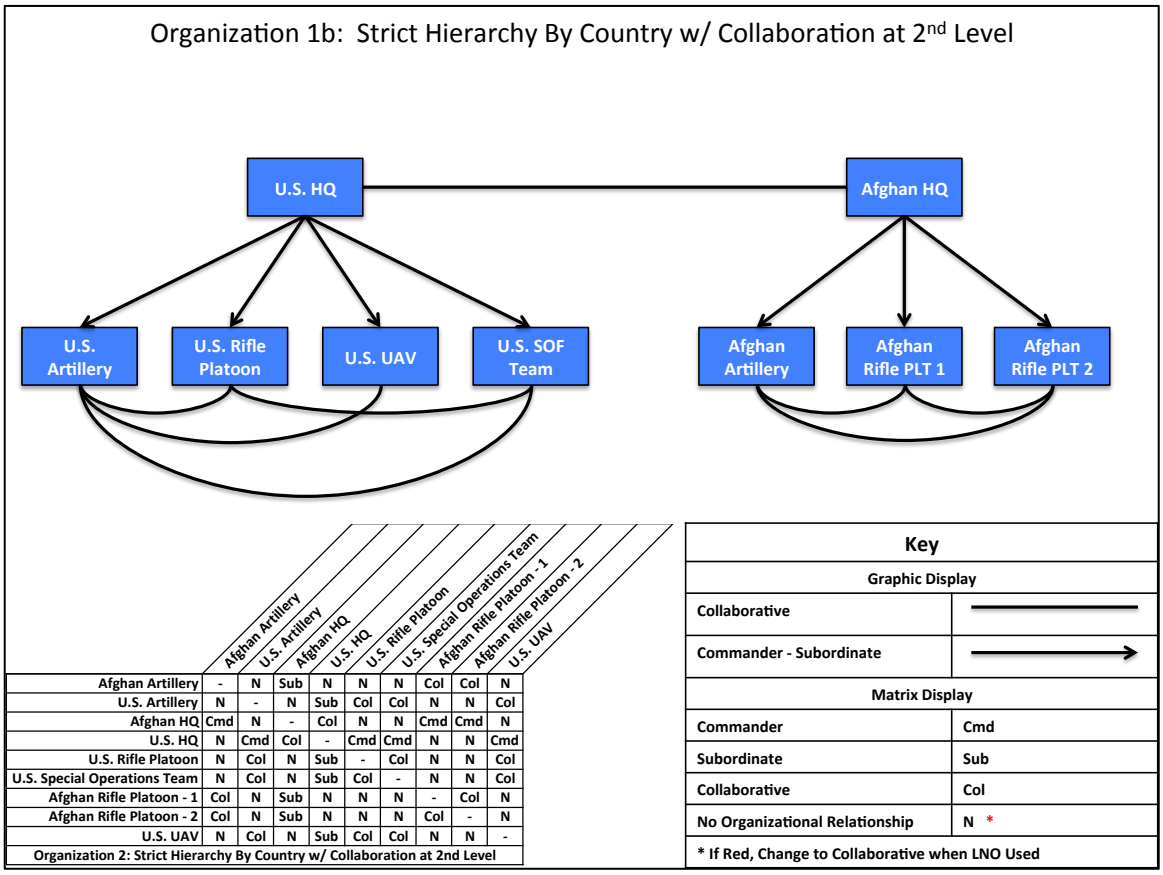


	Afghan Artillery	U.S. Artillery	Afghan HQ	U.S. HQ	U.S. Rifle Platoon	U.S. Special Operations Team	Afghan Rifle Platoon - 1	Afghan Rifle Platoon - 2	U.S. UAV
Afghan Artillery	-	N	Sub	N	N	N	N	N	N
U.S. Artillery	N	-	N	Sub	N	N	N	N	N
Afghan HQ	Cmd	N	-	Col	N	N	Cmd	Cmd	N
U.S. HQ	N	Cmd	Col	-	Cmd	Cmd	N	N	Cmd
U.S. Rifle Platoon	N	N	N	Sub	-	N	N	N	N
U.S. Special Operations Team	N	N	N	Sub	N	-	N	N	N
Afghan Rifle Platoon - 1	N	N	Sub	N	N	N	-	N	N
Afghan Rifle Platoon - 2	N	N	Sub	N	N	N	N	-	N
U.S. UAV	N	N	N	Sub	N	N	N	N	-

Organization 1: Strict Hierarchy By Country

Key	
Graphic Display	
Collaborative	———
Commander - Subordinate	———→
Matrix Display	
Commander	Cmd
Subordinate	Sub
Collaborative	Col
No Organizational Relationship	N *
* If Red, Change to Collaborative when LNO Used	

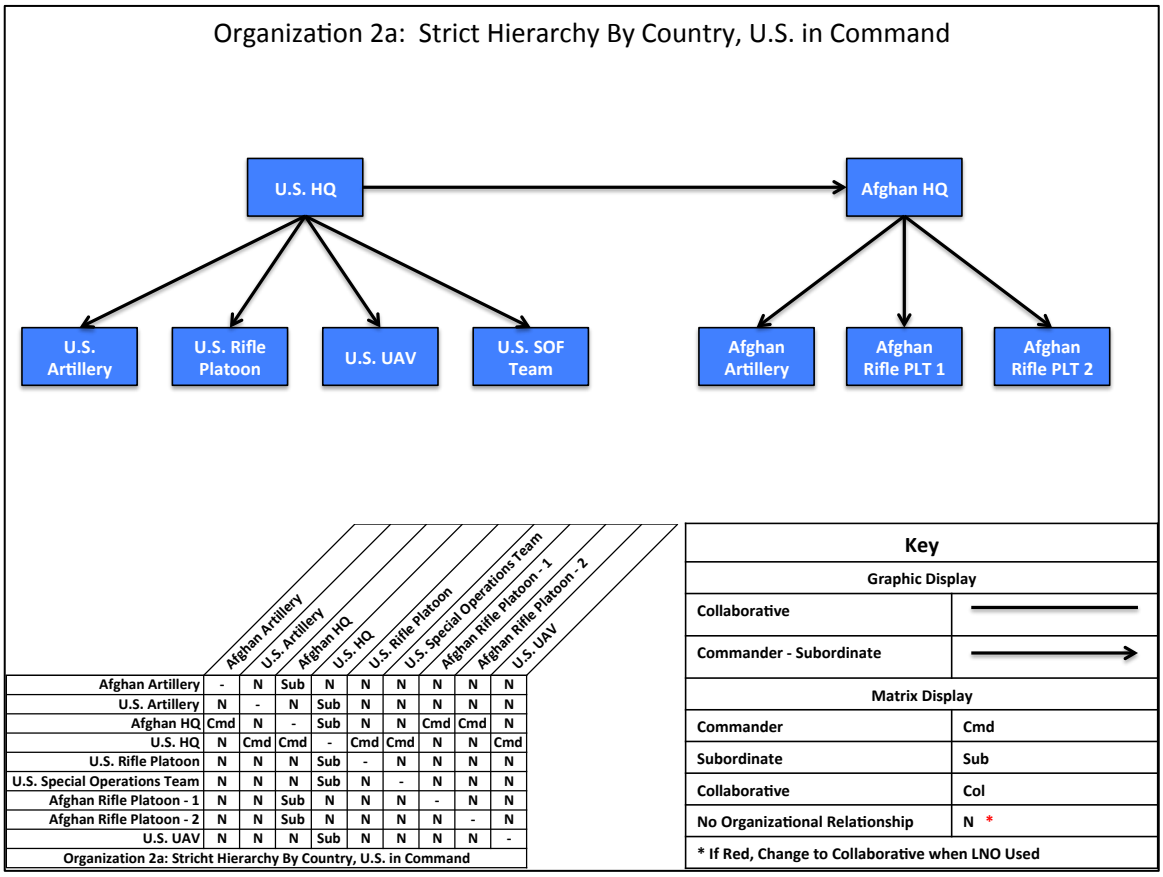
Figure 75. Organization 1a



	Afghan Artillery	U.S. Artillery	Afghan HQ	U.S. HQ	U.S. Rifle Platoon	U.S. Special Operations Team	Afghan Rifle Platoon - 1	Afghan Rifle Platoon - 2	U.S. UAV
Afghan Artillery	-	N	Sub	N	N	N	Col	Col	N
U.S. Artillery	N	-	N	Sub	Col	Col	N	N	Col
Afghan HQ	Cmd	N	-	Col	N	N	Cmd	Cmd	N
U.S. HQ	N	Cmd	Col	-	Cmd	Cmd	N	N	Cmd
U.S. Rifle Platoon	N	Col	N	Sub	-	Col	N	N	Col
U.S. Special Operations Team	N	Col	N	Sub	Col	-	N	N	Col
Afghan Rifle Platoon - 1	Col	N	Sub	N	N	N	-	Col	N
Afghan Rifle Platoon - 2	Col	N	Sub	N	N	N	Col	-	N
U.S. UAV	N	Col	N	Sub	Col	Col	N	N	-

Organization 2: Strict Hierarchy By Country w/ Collaboration at 2nd Level

Figure 76. Organization 1b



	Afghan Artillery	U.S. Artillery	Afghan HQ	U.S. HQ	U.S. Rifle Platoon	U.S. Special Operations Team	Afghan Rifle Platoon - 1	Afghan Rifle Platoon - 2	U.S. UAV
Afghan Artillery	-	N	Sub	N	N	N	N	N	N
U.S. Artillery	N	-	N	Sub	N	N	N	N	N
Afghan HQ	Cmd	N	-	Sub	N	N	Cmd	Cmd	N
U.S. HQ	N	Cmd	Cmd	-	Cmd	Cmd	N	N	Cmd
U.S. Rifle Platoon	N	N	N	Sub	-	N	N	N	N
U.S. Special Operations Team	N	N	N	Sub	N	-	N	N	N
Afghan Rifle Platoon - 1	N	N	Sub	N	N	N	-	N	N
Afghan Rifle Platoon - 2	N	N	Sub	N	N	N	N	-	N
U.S. UAV	N	N	N	Sub	N	N	N	N	-

Organization 2a: Stricht Hierarchy By Country, U.S. in Command

Figure 77. Organization 2a

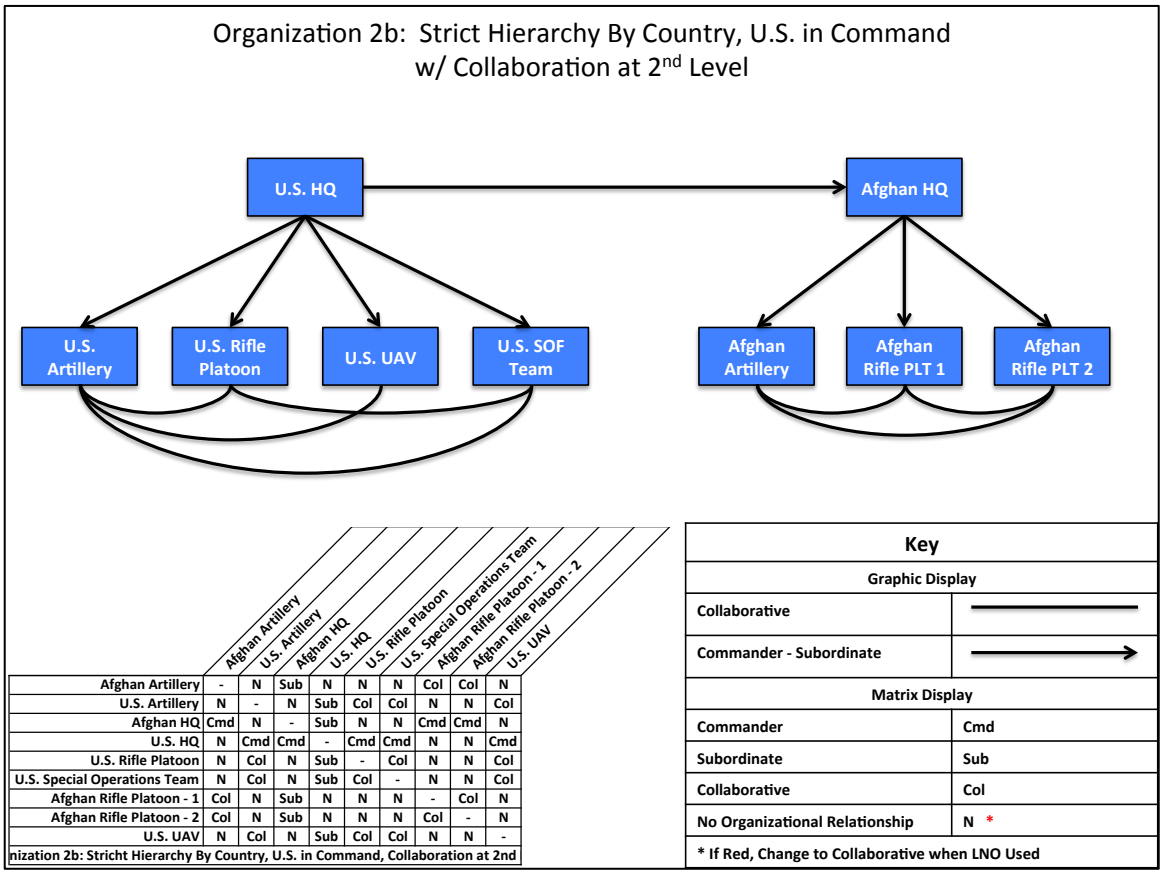
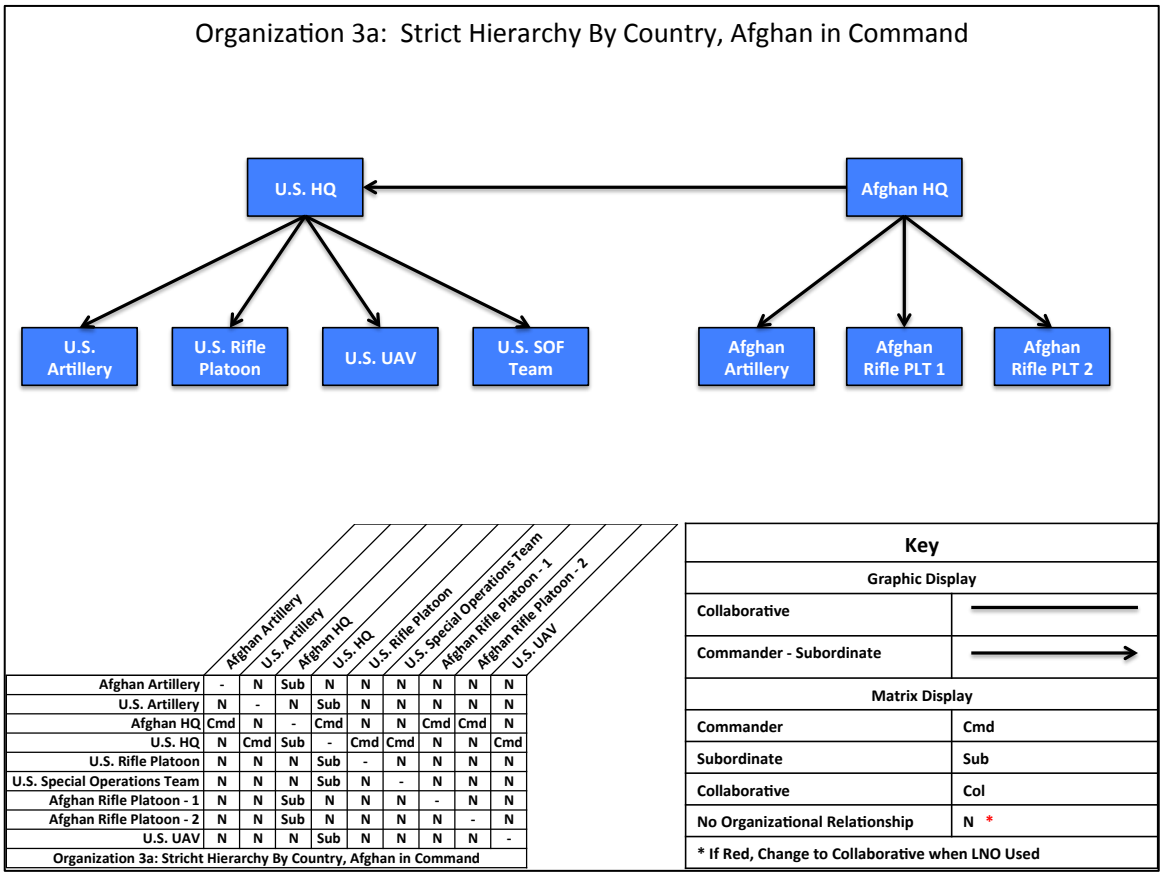


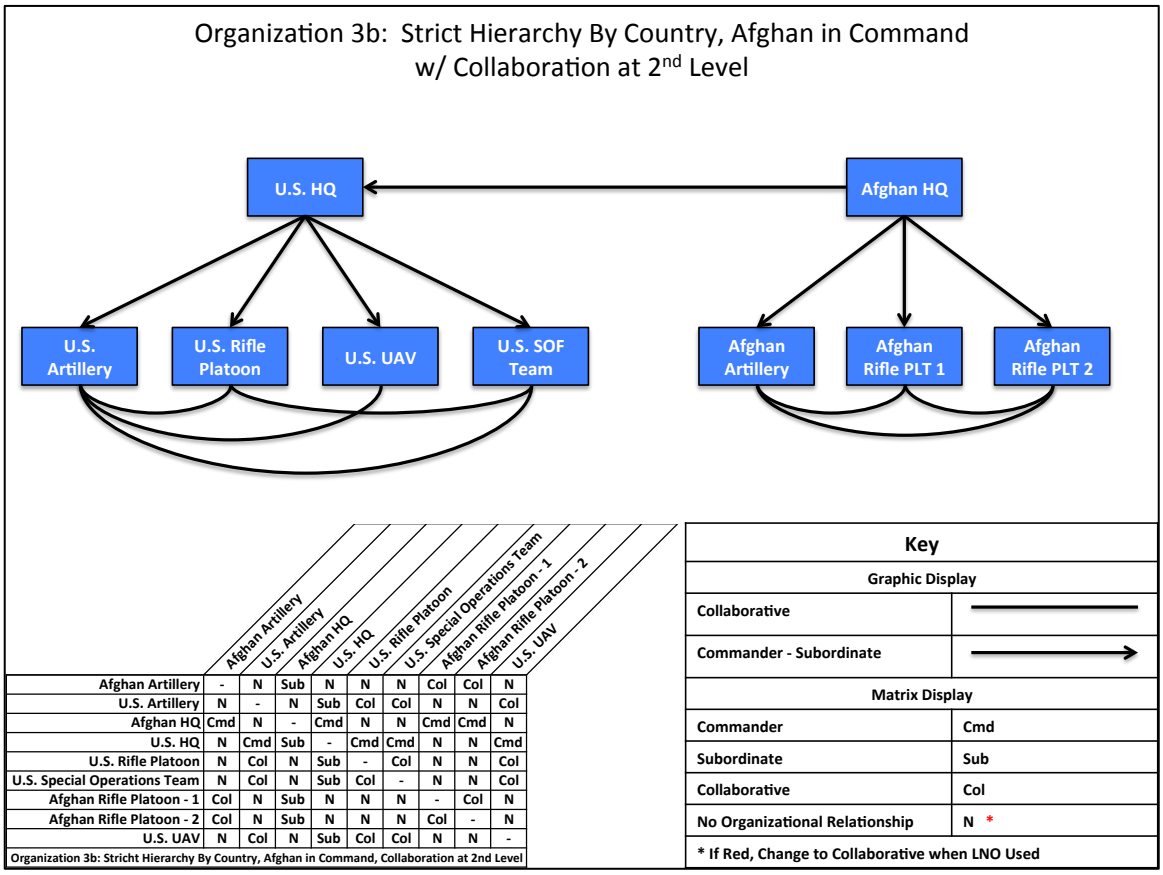
Figure 78. Organization 2b



	Afghan Artillery	U.S. Artillery	Afghan HQ	U.S. HQ	U.S. Rifle Platoon	U.S. Special Operations Team	Afghan Rifle Platoon - 1	Afghan Rifle Platoon - 2	U.S. UAV
Afghan Artillery	-	N	Sub	N	N	N	N	N	N
U.S. Artillery	N	-	N	Sub	N	N	N	N	N
Afghan HQ	Cmd	N	-	Cmd	N	N	Cmd	Cmd	N
U.S. HQ	N	Cmd	Sub	-	Cmd	Cmd	N	N	Cmd
U.S. Rifle Platoon	N	N	N	Sub	-	N	N	N	N
U.S. Special Operations Team	N	N	N	Sub	N	-	N	N	N
Afghan Rifle Platoon - 1	N	N	Sub	N	N	N	-	N	N
Afghan Rifle Platoon - 2	N	N	Sub	N	N	N	N	-	N
U.S. UAV	N	N	N	Sub	N	N	N	N	-

Organization 3a: Stricht Hierarchy By Country, Afghan in Command

Figure 79. Organization 3a

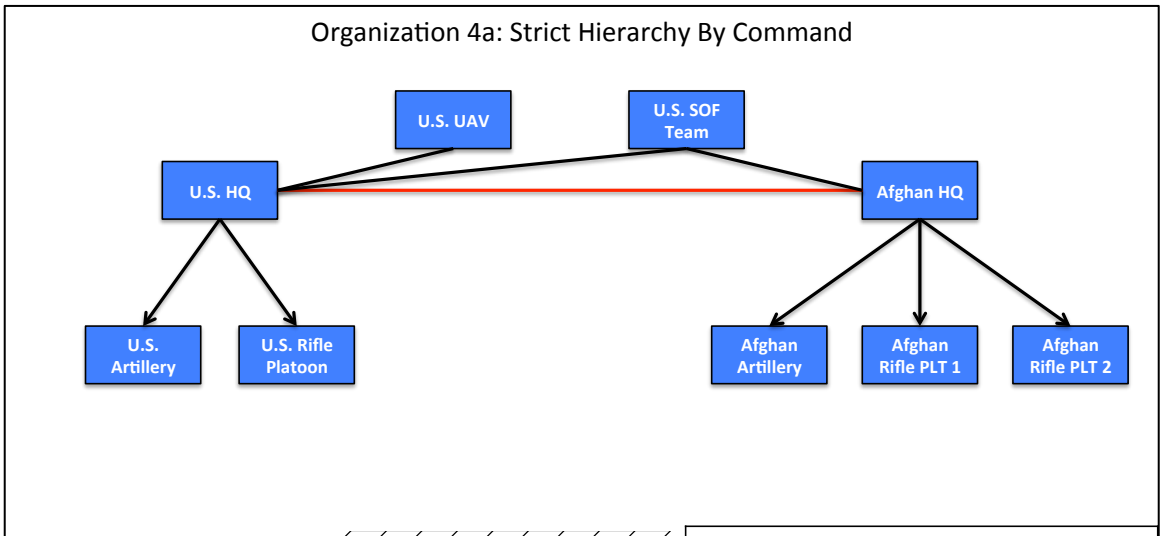


	Afghan Artillery	U.S. Artillery	Afghan HQ	U.S. HQ	U.S. Rifle Platoon	U.S. Special Operations Team	Afghan Rifle Platoon - 1	Afghan Rifle Platoon - 2	U.S. UAV
Afghan Artillery	-	N	Sub	N	N	N	Col	Col	N
U.S. Artillery	N	-	N	Sub	Col	Col	N	N	Col
Afghan HQ	Cmd	N	-	Cmd	N	N	Cmd	Cmd	N
U.S. HQ	N	Cmd	Sub	-	Cmd	Cmd	N	N	Cmd
U.S. Rifle Platoon	N	Col	N	Sub	-	Col	N	N	Col
U.S. Special Operations Team	N	Col	N	Sub	Col	-	N	N	Col
Afghan Rifle Platoon - 1	Col	N	Sub	N	N	N	-	Col	N
Afghan Rifle Platoon - 2	Col	N	Sub	N	N	N	Col	-	N
U.S. UAV	N	Col	N	Sub	Col	Col	N	N	-

Organization 3b: Stricht Hierarchy By Country, Afghan in Command, Collaboration at 2nd Level

Key	
Graphic Display	
Collaborative	———
Commander - Subordinate	———→
Matrix Display	
Commander	Cmd
Subordinate	Sub
Collaborative	Col
No Organizational Relationship	N *
* If Red, Change to Collaborative when LNO Used	

Figure 80. Organization 3b

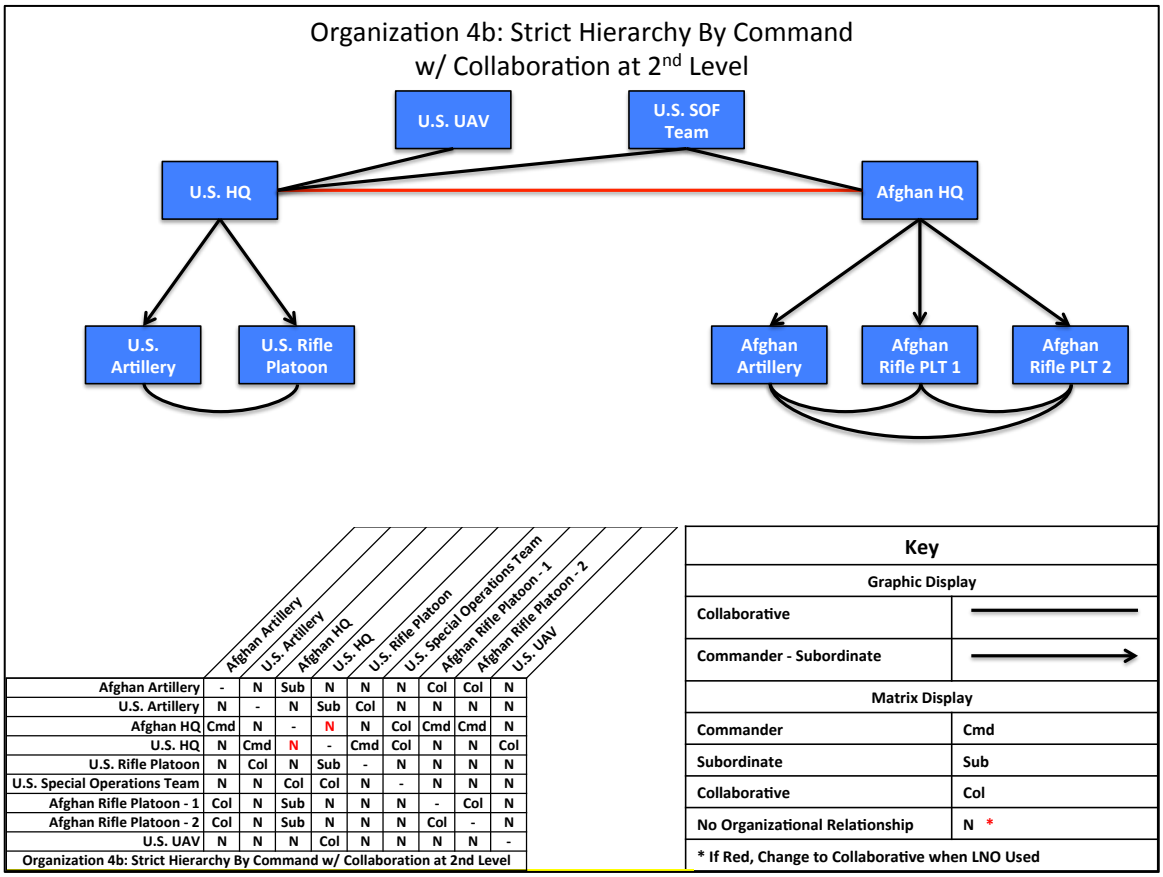


	Afghan Artillery	U.S. Artillery	Afghan HQ	U.S. HQ	U.S. Rifle Platoon	U.S. Special Operations Team	Afghan Rifle Platoon - 1	Afghan Rifle Platoon - 2	U.S. UAV
Afghan Artillery	-	N	Sub	N	N	N	N	N	N
U.S. Artillery	N	-	N	Sub	N	N	N	N	N
Afghan HQ	Cmd	N	-	N	N	Col	Cmd	Cmd	N
U.S. HQ	N	Cmd	N	-	Cmd	Col	N	N	Col
U.S. Rifle Platoon	N	N	N	Sub	-	N	N	N	N
U.S. Special Operations Team	N	N	Col	Col	N	-	N	N	N
Afghan Rifle Platoon - 1	N	N	Sub	N	N	N	-	N	N
Afghan Rifle Platoon - 2	N	N	Sub	N	N	N	N	-	N
U.S. UAV	N	N	N	Col	N	N	N	N	-

Organization 4a: Strict Hierarchy By Command

Key	
Graphic Display	
Collaborative	———
Commander - Subordinate	———→
Matrix Display	
Commander	Cmd
Subordinate	Sub
Collaborative	Col
No Organizational Relationship	N *
* If Red, Change to Collaborative when LNO Used	

Figure 81. Organization 4a



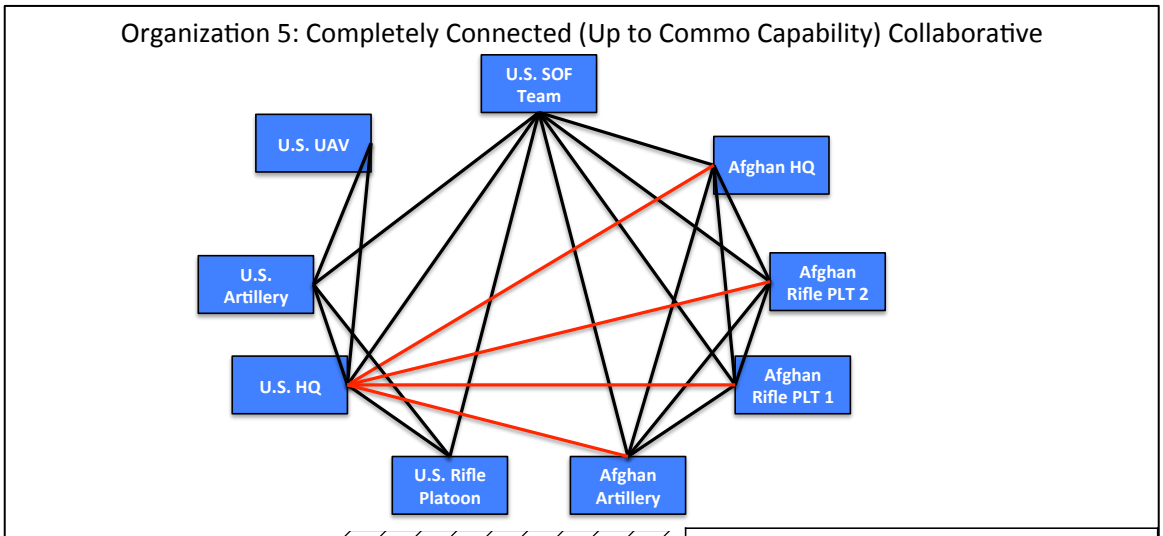
	Afghan Artillery	U.S. Artillery	Afghan HQ	U.S. HQ	U.S. Rifle Platoon	U.S. Special Operations Team	Afghan Rifle Platoon - 1	Afghan Rifle Platoon - 2	U.S. UAV
Afghan Artillery	-	N	Sub	N	N	N	Col	Col	N
U.S. Artillery	N	-	N	Sub	Col	N	N	N	N
Afghan HQ	Cmd	N	-	N	N	Col	Cmd	Cmd	N
U.S. HQ	N	Cmd	N	-	Cmd	Col	N	N	Col
U.S. Rifle Platoon	N	Col	N	Sub	-	N	N	N	N
U.S. Special Operations Team	N	N	Col	Col	N	-	N	N	N
Afghan Rifle Platoon - 1	Col	N	Sub	N	N	N	-	Col	N
Afghan Rifle Platoon - 2	Col	N	Sub	N	N	N	Col	-	N
U.S. UAV	N	N	N	Col	N	N	N	N	-

Organization 4b: Strict Hierarchy By Command w/ Collaboration at 2nd Level

Key	
Graphic Display	
Collaborative	———
Commander - Subordinate	———→
Matrix Display	
Commander	Cmd
Subordinate	Sub
Collaborative	Col
No Organizational Relationship	N *

* If Red, Change to Collaborative when LNO Used

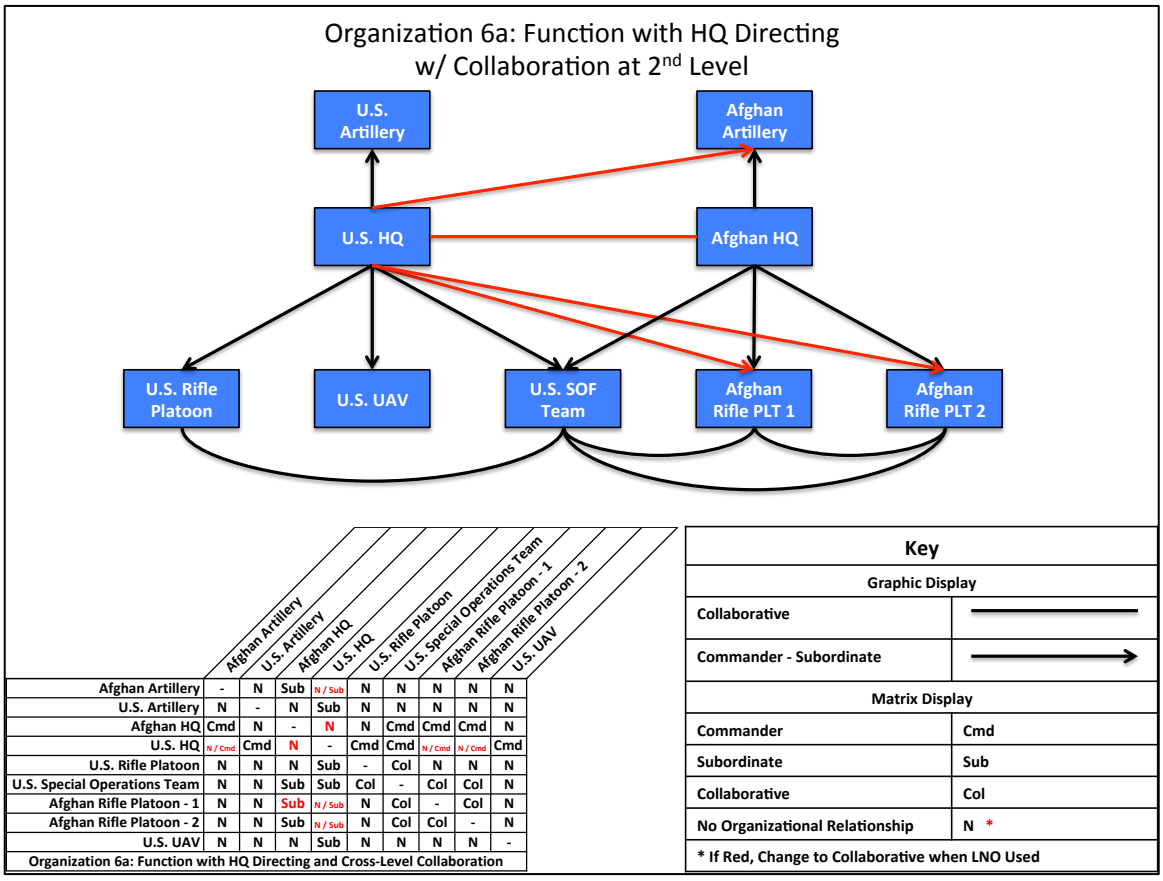
Figure 82. Organization 4b



	Afghan Artillery	U.S. Artillery	Afghan HQ	U.S. HQ	U.S. Rifle Platoon	U.S. Special Operations Team	Afghan Rifle Platoon - 1	Afghan Rifle Platoon - 2	U.S. UAV
Afghan Artillery	-	N	N	N	N	Col	Col	Col	N
U.S. Artillery	N	-	N	Col	Col	Col	N	N	Col
Afghan HQ	Col	N	-	N	N	Col	Col	Col	N
U.S. HQ	N	Col	N	-	Col	Col	N	N	Col
U.S. Rifle Platoon	N	Col	N	Col	-	Col	N	N	N
U.S. Special Operations Team	Col	Col	Col	Col	Col	-	Col	Col	N
Afghan Rifle Platoon - 1	Col	N	Col	N	N	Col	-	Col	N
Afghan Rifle Platoon - 2	Col	N	Col	N	N	Col	Col	-	N
U.S. UAV	N	Col	N	Col	N	N	N	N	-
Organization 5: Completely Connected (Up to Commo Capability) Collaborative									

Key	
Graphic Display	
Collaborative	———
Commander - Subordinate	———→
Matrix Display	
Commander	Cmd
Subordinate	Sub
Collaborative	Col
No Organizational Relationship	N *
* If Red, Change to Collaborative when LNO Used	

Figure 83. Organization 5

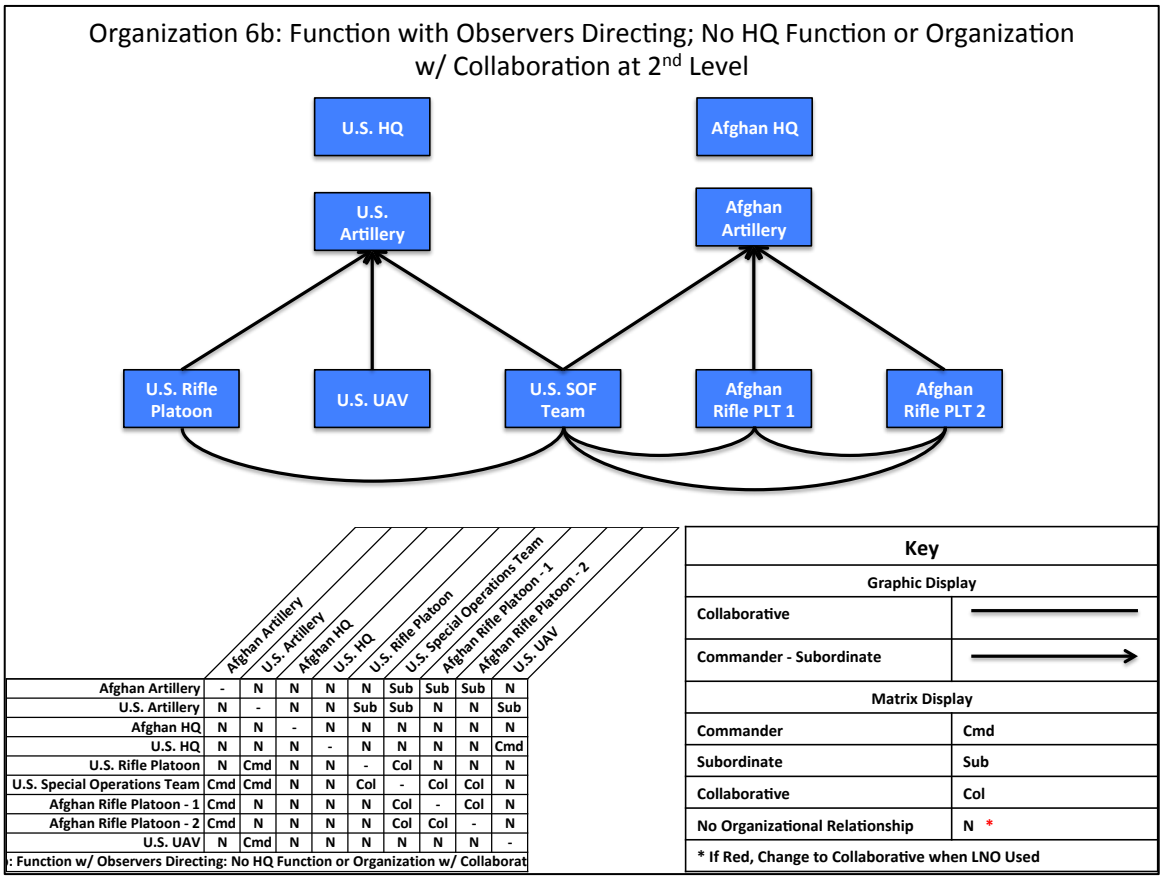


	Afghan Artillery	U.S. Artillery	Afghan HQ	U.S. HQ	U.S. Rifle Platoon	U.S. Special Operations Team	Afghan Rifle Platoon - 1	Afghan Rifle Platoon - 2	U.S. UAV
Afghan Artillery	-	N	Sub	N / Sub	N	N	N	N	N
U.S. Artillery	N	-	N	Sub	N	N	N	N	N
Afghan HQ	Cmd	N	-	N	N	Cmd	Cmd	Cmd	N
U.S. HQ	N / Cmd	Cmd	N	-	Cmd	Cmd	N / Cmd	N / Cmd	Cmd
U.S. Rifle Platoon	N	N	N	Sub	-	Col	N	N	N
U.S. Special Operations Team	N	N	Sub	Sub	Col	-	Col	Col	N
Afghan Rifle Platoon - 1	N	N	Sub	N / Sub	N	Col	-	Col	N
Afghan Rifle Platoon - 2	N	N	Sub	N / Sub	N	Col	Col	-	N
U.S. UAV	N	N	N	Sub	N	N	N	N	-

Organization 6a: Function with HQ Directing and Cross-Level Collaboration

Key	
Graphic Display	
Collaborative	———
Commander - Subordinate	———→
Matrix Display	
Commander	Cmd
Subordinate	Sub
Collaborative	Col
No Organizational Relationship	N *
* If Red, Change to Collaborative when LNO Used	

Figure 84. Organization 6a



	Afghan Artillery	U.S. Artillery	Afghan HQ	U.S. HQ	U.S. Rifle Platoon	U.S. Special Operations Team	Afghan Rifle Platoon - 1	Afghan Rifle Platoon - 2	U.S. UAV
Afghan Artillery	-	N	N	N	N	Sub	Sub	Sub	N
U.S. Artillery	N	-	N	N	Sub	Sub	N	N	Sub
Afghan HQ	N	N	-	N	N	N	N	N	N
U.S. HQ	N	N	N	-	N	N	N	N	Cmd
U.S. Rifle Platoon	N	Cmd	N	N	-	Col	N	N	N
U.S. Special Operations Team	Cmd	Cmd	N	N	Col	-	Col	Col	N
Afghan Rifle Platoon - 1	Cmd	N	N	N	N	Col	-	Col	N
Afghan Rifle Platoon - 2	Cmd	N	N	N	N	Col	Col	-	N
U.S. UAV	N	Cmd	N	N	N	N	N	N	-

Figure 85. Organization 6b

Acceptable Relationships

Basic Rules / Reasoning

- All will command or collaborate except as listed below
  - Units willing to subordinate highlighted in yellow
- SOF Team will work in any manner with any system
- U.S. HQ will not subordinate to anyone, but command or collaborate with anyone
- Afghan HQ will only subordinate to U.S. HQ or SOF
- Afghan Line Elements will subordinate to U.S. units or Afghan HQ, but not each other
- U.S. UAV will only work with U.S. systems

	Afghan Artillery	U.S. Artillery	Afghan HQ	U.S. HQ	U.S. Rifle Platoon	U.S. Special Operations Team	Afghan Rifle Platoon - 1	Afghan Rifle Platoon - 2	U.S. UAV
Afghan Artillery	-	Cmd, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Col, N	Cmd, Col, N	Cmd, Sub, Col, N
U.S. Artillery	Cmd, Col, N	-	Cmd, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Col, N	Cmd, Col, N	Cmd, Sub, Col, N
Afghan HQ	Cmd, Col, N	Cmd, Col, N	-	Cmd, Sub, Col, N	Cmd, Col, N	Cmd, Sub, Col, N	Cmd, Col, N	Cmd, Col, N	Cmd, Col, N
U.S. HQ	Cmd, Col, N	Cmd, Col, N	Cmd, Col, N	-	Cmd, Col, N	Cmd, Col, N	Cmd, Col, N	Cmd, Col, N	Cmd, Col, N
U.S. Rifle Platoon	Cmd, Col, N	Cmd, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	-	Cmd, Sub, Col, N	Cmd, Col, N	Cmd, Col, N	Cmd, Col, N
U.S. Special Operations Team	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	-	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N
Afghan Rifle Platoon - 1	Cmd, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	-	Cmd, Col, N	Cmd, Sub, Col, N
Afghan Rifle Platoon - 2	Cmd, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Col, N	-	Cmd, Sub, Col, N
U.S. UAV	N	Cmd, Sub, Col, N	N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	Cmd, Sub, Col, N	N	N	-

Acceptable Relationships: Cmd = Command; Sub = Subordinate; Col = Collaborative; N = No Relationship

Figure 86. Acceptable Organization Chart

C. INDIRECT FIRE OPERATIONAL SIMULATION

1. Methods and Notes

For this example, the author used MATLAB to define the IDF-SoS operational and cost models. This was both for convenience and control. MATLAB is readily available and has extensive network science packages. Furthermore, by using MATLAB for both the operational and cost models, the author was able to make explicit the utility of the SoS-TDM. In future situations, an engineer may use any general-purpose programming language (e.g., JAVA, Python), particularly ones that have pre-built network science routines. For the SoS operational modeling and analysis, any of a number of models are appropriate depending up the desired performance measures (e.g., ABM for operational performance such as AnyLogic or MANA, or cost models such as COSYSMO or CoCoMo II). Finally, note that the scenario was for academic purposes only. The capabilities of all systems are notional, based upon reasonable judgment and open source information.

2. Indirect Fire Definition

Indirect fires are "Fire delivered at a target which cannot be seen by the aimer" (NATO 2015, 2-I-3). Note that the term fire is used in the common military terminology, e.g. "fires – The use of weapon systems to create a specific lethal or nonlethal effect on a target" (ADRP 1–02 2015, 1–39). In this example, the prescribed effect on a target is destruction. This is defined as: "Destroy – A tactical mission task that physically renders an enemy force combat-ineffective until it is reconstituted. Alternatively, to destroy a combat system is to damage it so badly that it cannot perform any function or be restored to usable condition without being entirely rebuilt" (ADRP 1–02 2015, 1–28).

Typically, indirect fire is indirect (i.e., the shooter cannot see the target) for one of two reasons: 1) the distance from the shooter to the target is so great that the shooter cannot see the target or 2) there is an obstacle between the target and the shooter. In either event, the laws of gravity govern the ballistics of the projectile as seen in Figure 87.

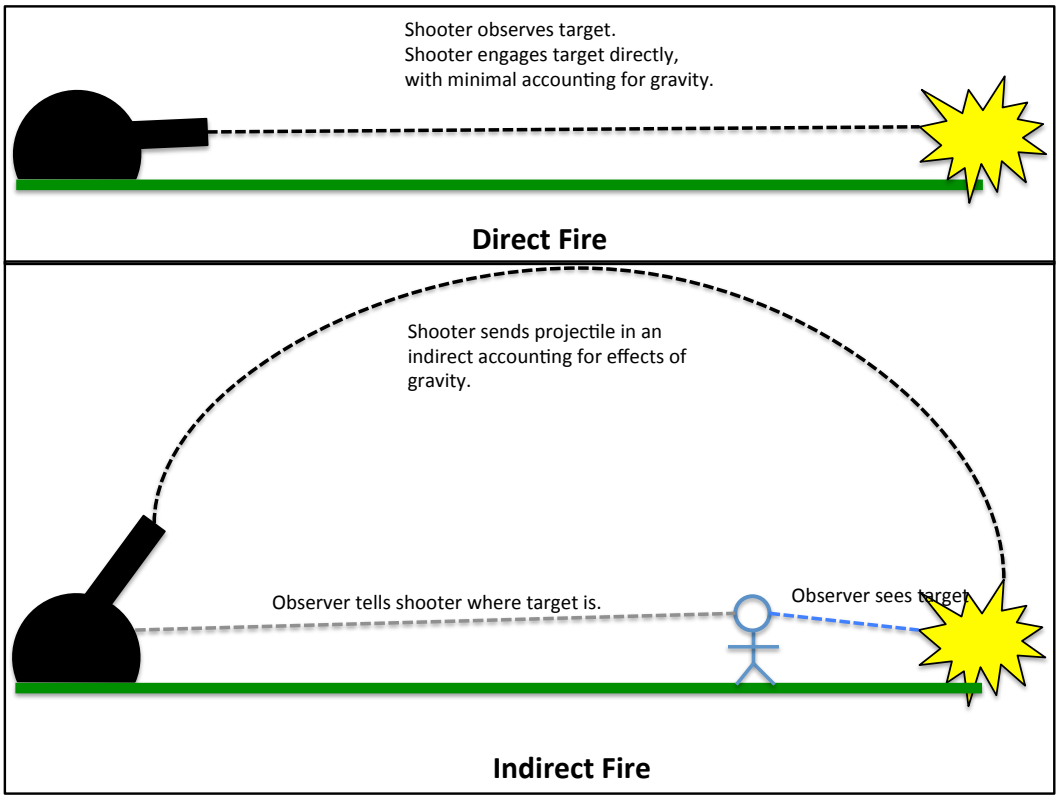


Figure 87. Direct versus Indirect Fire

By the strict definition of IDF, no observer is necessary; however, in order to have aimed fire, an observer must see the target and relay information to the shooter. For this example, it is assumed that only aimed IDF is allowable. This leads to some basic requirements for an IDF SoS.

The first IDF SoS requirement is that there must be a system or collection of systems that provide an "effect on a target." In this case, the effect is destruction—imparting damage to such an extent that the target is no longer functional. This means the SoS must have a system that sends a projectile. Typically, IDF systems in the military are artillery, mortars, rockets, and missiles of various sorts.

The second IDF SoS requirement is that there is a system or collection of systems that observe and provide information about the target. This is distinct from any shooter system as, by definition, the shooter, in an IDF system, cannot observe the target.

The third IDF SoS requirement is that information about the target is communicated, and possibly processed, between the aforementioned systems. The basic information of target location, description, and other factors must be relayed from the observer to the shooter. Furthermore, in order for a shooter to make an informed choice, it may need to know target priorities and the locations of other systems in the area (target deconfliction).

An IDF SoS is one that integrates the capabilities of observers and shooters via communication and information processing to provide aimed indirect fire on enemy targets.

D. IDF-SOS OPERATIONAL MODEL

For this model, the scenario, or "Area of Operations" is defined as a series of vertical "lanes" that divide a map. Each one of these lanes is a target area, in which all shooters have the ability to engage and all observers have the potential to see. In the program, this is defined as a vector in which each entry corresponds to a lane as depicted in Figure 88.

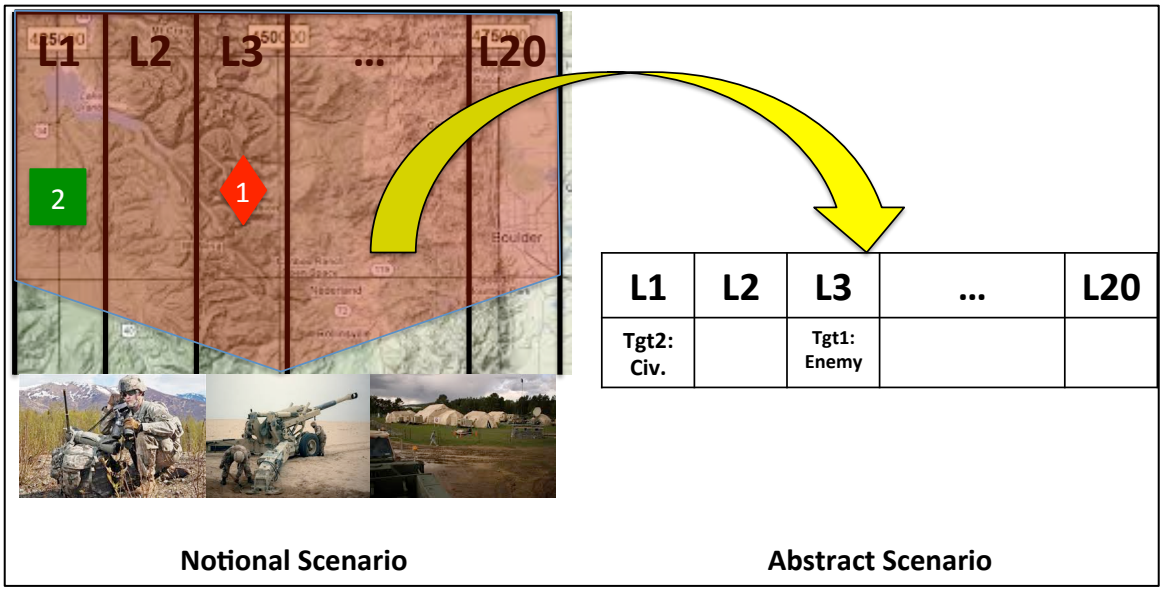


Figure 88. Area of Operations and Its Abstraction

Time in the scenario is represented by a time step, notionally one minute long. During each time step, every entity views its environment, makes decisions, and then acts simultaneously. Targets appear during each time step according to a Poisson distribution with a mean of one. Each target is assigned a location by a random uniform distribution across the map; each target is equally probably a civilian or enemy target; each target is given a pre-determined "presentation time" – how long it exists in the scenario, this is determined by a Poisson distribution, with a mean of 7 (i.e., 7 minutes).

As location is immaterial in this scenario (i.e., it is abstracted away, when we assume that all observers can observe any location equally well, and all shooters can engage any location equally well), the systems in the SoS are not assigned a location.

A general outline of how the algorithm that defined the IDF-SoS ABM ran follows. This is not intended to be a formal demonstration of the simulation; rather, it is a broad overview.

- **Target Creation and Destruction:** Targets are created randomly according to a Poisson distribution; targets are removed if they have met their "presentation time."

- **External Observations:** Observer systems make observations relative to what targets are presented and their P_{detect} seen in Figure 44. If an observer detects a target, it locates it according to its P_{locate} and classifies it as civilian or enemy according to its $P_{classify}$. Note that deconflicters or shooter systems do not make external observations, they receive all environmental input from communications received from the SoS.

- **Internal Observations:** Each system "reads its messages" received, for that time step, from other systems in the SoS. It classifies messages in one of two ways—as messages to pass along or as messages intended for itself.

- **Message Passing:** Each system passes messages it received, but not intended for it, to the intended recipient, or, if the system and the recipient are not organizationally connected, to another system to which it is organizationally connected and is on the shortest path to the recipient.

  For example, if the current system is the SOF Team and the recipient is the U.S. Artillery; if the organization is one in which the SOF Team and U.S. Artillery are connected (e.g., Organization 1b), the SOF team sends that message directly to the U.S. Artillery. If, however, they are not organizationally connected (e.g., Organization 1a), the SOF team chooses

a recipient on the shortest path to the end system, in this case, that would be the U.S. Headquarters (that is, the message would go from the SOF team to the U.S. Headquarters to the U.S. Artillery).

The sender attempts to maintain the same communications system for the forwarded message as it was received on (e.g., in the example with the SOF Team, if the message was sent on U.S. FM, the SOF Team forwards it on U.S. FM). If, however, the sender and next recipient do not share that communications platform, the sender must "translate" the message. This takes an additional time step and the message will be received in two time steps.

Furthermore, each message is assessed as delivered or not according to the probability that the message is received and understood for that communications system as seen in Table 12.

- **Message Reading:** Each included system then "reads" its messages. Messages are of one of two forms, either a Call for Fire (CFF) or a Request for Information (RFI). Observers can receive RFI and deconflicters can receive CFF. How each system responds to the message depends upon the system itself, the organization, and the process. In general, the process is as follows:

- **Shooters:** The shooters read their messages and determine if they have one from a commander. If so, they prioritize those messages.

    If they do have an RFI, which requests what information is known about a single location (e.g., how many civilian or enemy targets there are), they assess their worldview, which is a composite of their observations as far back as their "memory" allows.

    If the systems do not have RFI to respond to, they develop calls for fire based upon their worldview. They choose possible targets dependent upon the ROE (either maximize the difference between enemy and civilian at a location or maximize locations without civilians, but with enemy) and then choose a recipient.

    If the process requires a deconflicter, they send to the deconflicter that is also their commander (if one exists) or otherwise randomly choose between the two. The actual message is sent in a manner similar to how it is described in the "message passing section."

- **Deconflicters:** Deconflicters develop their worldview based upon the various CFF they receive from observers. They do this in an additive manner. For example, if the "U.S. Rifle Platoon" and "UAV" both see an enemy target at location one and send that to the "U.S. Headquarters." The "U.S. Headquarters" assesses this to be two (independent) observations of

an enemy. Each headquarters' worldview is based upon how long their memory is.

The deconflicters then develop a new CFF based upon the ROE and send it to a shooter, with a priority on a subordinate. Note that in processes that do not require a deconflicter, even if one is included, it may not "have much to do" as systems will not send them CFF directly; it will only pass messages.

- **Shooters:** Shooters receive CFF either directly from observers in the relevant processes or from deconflicters.

  If the process includes deconflicters, shooters focus solely on shooting and simply prioritize shooting targets from their commanders, up to the max number of shots allowed.

  Otherwise, shooters must decide upon which CFF from observers to act on. They prioritize those from their commanders and then make a choice in a similar manner as described for observers, although it is more limited as shooters have less "memory."

- **Shots Fired, Damage Assessed:** Once all systems have made their decisions, if the shooters made any shots, the effects of those shots are assessed.

  Each shot lands at its location according to the shooter's P_{hit}; if a shot misses, it lands in the location immediately left or right of the target area with equal probability.

  Each target at the impact location is then assessed for damage according to the P_{kill}. If a target is "killed" it is considered destroyed and removed from the simulation.

- **Iteration:** These steps are iterated for the length of the scenario. At the end of the scenario, the results are tallied, the number of enemy targets that appeared, how many were hit, and the same for the civilian targets. These results provide the MOEs PTD and PCD and are the outputs for that design point.

E. IDF-SOS COST MODEL

The IDF SoS cost model is a deterministic formula. The cost of an SoS is a function of the sum of the cost of each system in Table 20. Each organizational relationship included in an SoS cost a varying amount as indicated in Table 20. Finally, the cost of a chosen process was a function of the number of operational activities required.

Table 20. SoS Cost Table

System, Relationship, or Process	Cost
Afghan Artillery	$20,000
U.S. Artillery	$100,000
Afghan Headquarters	$50,000
U.S. Headquarters	$200,000
U.S. Rifle Platoon	$50,000
Special Operations Team	$120,000
Afghan Rifle Platoon – 1	$10,000
Afghan Rifle Platoon – 2	$10,000
UAV	$200,000
Afghan LNO	$50,000
Collaborative Relationship	15,000
Subordinate Relationship	$30,000
Command Relationship	$30,000
Operational Activity (Any Type)	$10,000

F. TRADESPACE EXPLORATION EXAMPLE

The tradespace of the IDF SoS is its design space, set of system attributes (PTD, PCD, and cost), and the bounds placed upon the allowable design parameters or system attributes as discussed in Section IV.E. Decision makers may vary the allowable bounds to assess what potential systems may satisfy their requirements. Note that this analysis did not include utility functions as these are a second source of subjectivity; however, it is fairly simple to modify the tradespace GUI to allow a user to define each utility function and their corresponding weights.

As an example, for the IDF SoS, there are a few features of the tradespace are useful to note. First, there is a general correlation between increasing PTD and PCD as seen in Figure 89. This goes against the desire to maximize PTD and minimize PCD. Decision-makers must contend with this trade-off while still considering cost requirements.

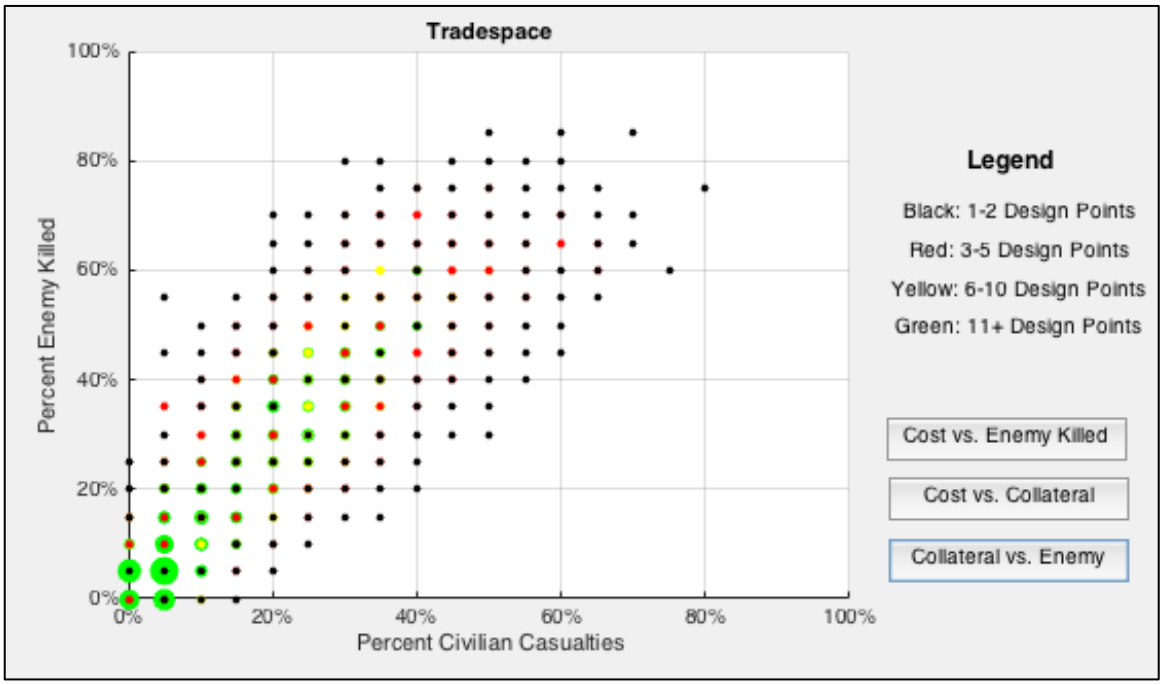


Figure 89. Percent Enemy Killed versus Percent Civilian Casualties, All Design Points

On the other hand, there is no apparent correlation with cost and PTD or PCD as seen in Figure 90.

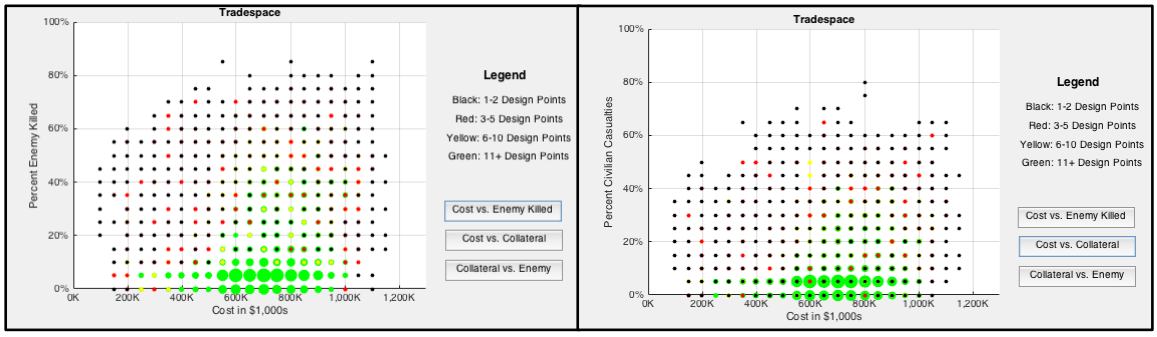


Figure 90. IDF-SoS, Cost versus PTD and Cost versus PCD

Accordingly, the decision-maker may begin to explore the tradespace per his internal values and requirements. For example, if a decision-maker prioritized minimizing collateral damage, he could only consider those design that exhibited no collateral damage and then compare PTD versus Cost as seen in Figure 91. Note that the

best of these design points, from a PTD perspective, is approximately PTD = 25% and cost approximately $600K.

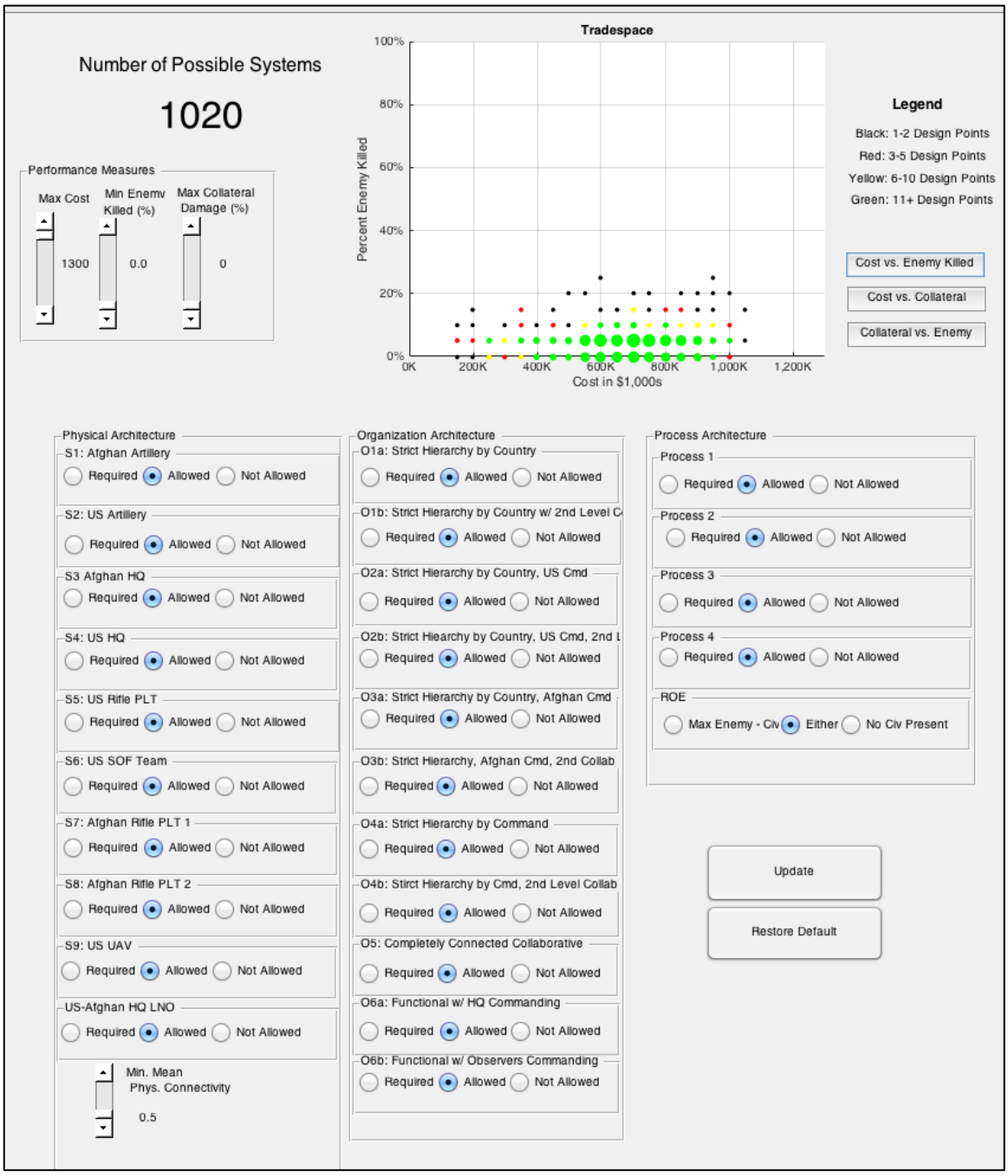


Figure 91. Design Points that Minimize Collateral Damage

If the decision-maker considers that 25% PTD is insufficient, and is willing to assume more risk with collateral damage, he may easily expand the set of potential designs to those that allow up to 10% PCD. This significantly increases the number of potential designs, increases the potential PTD to approximately 55%, and at a lower cost of approximately $150,000. This is seen in Figure 92.

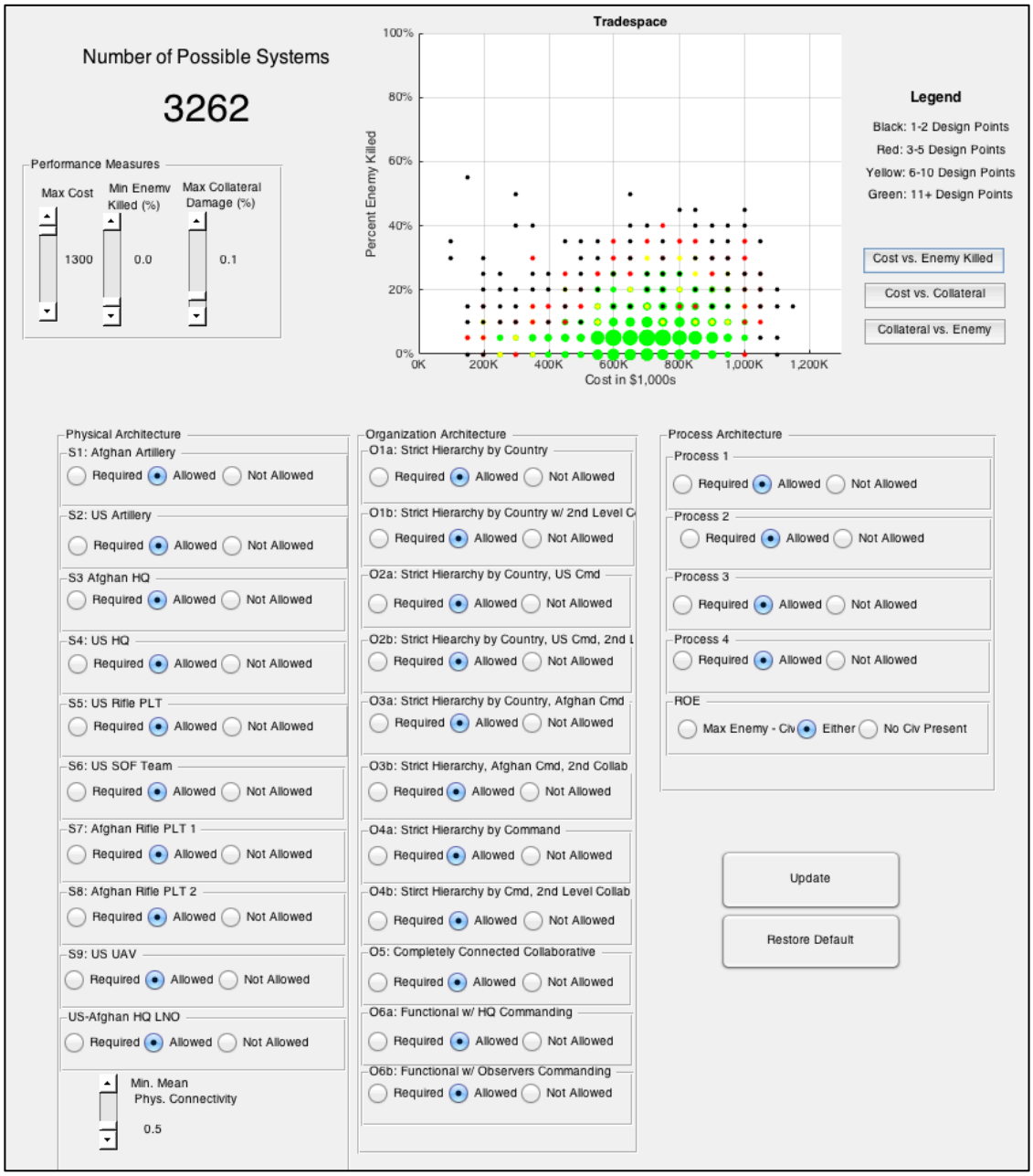


Figure 92. IDF-SoS Tradespace if 10% PCD is Allowable

Finally, if the decision-maker has concerns about including Afghan forces (perhaps for political reasons) and is tied to the idea of a hierarchical organization with the U.S. Army in control (again, perhaps for political reasons), but still wants less than 10% PCD, some of the previous results are not available. He may be able to achieve similar collateral damage results, but with reduced PTD (35% from 55%) and a higher cost ($500,000 versus $150,000). At a comparable collateral damage and cost (though still more expensive, at $250,000) he may achieve only 20% PTD. This is seen in Figure 93.

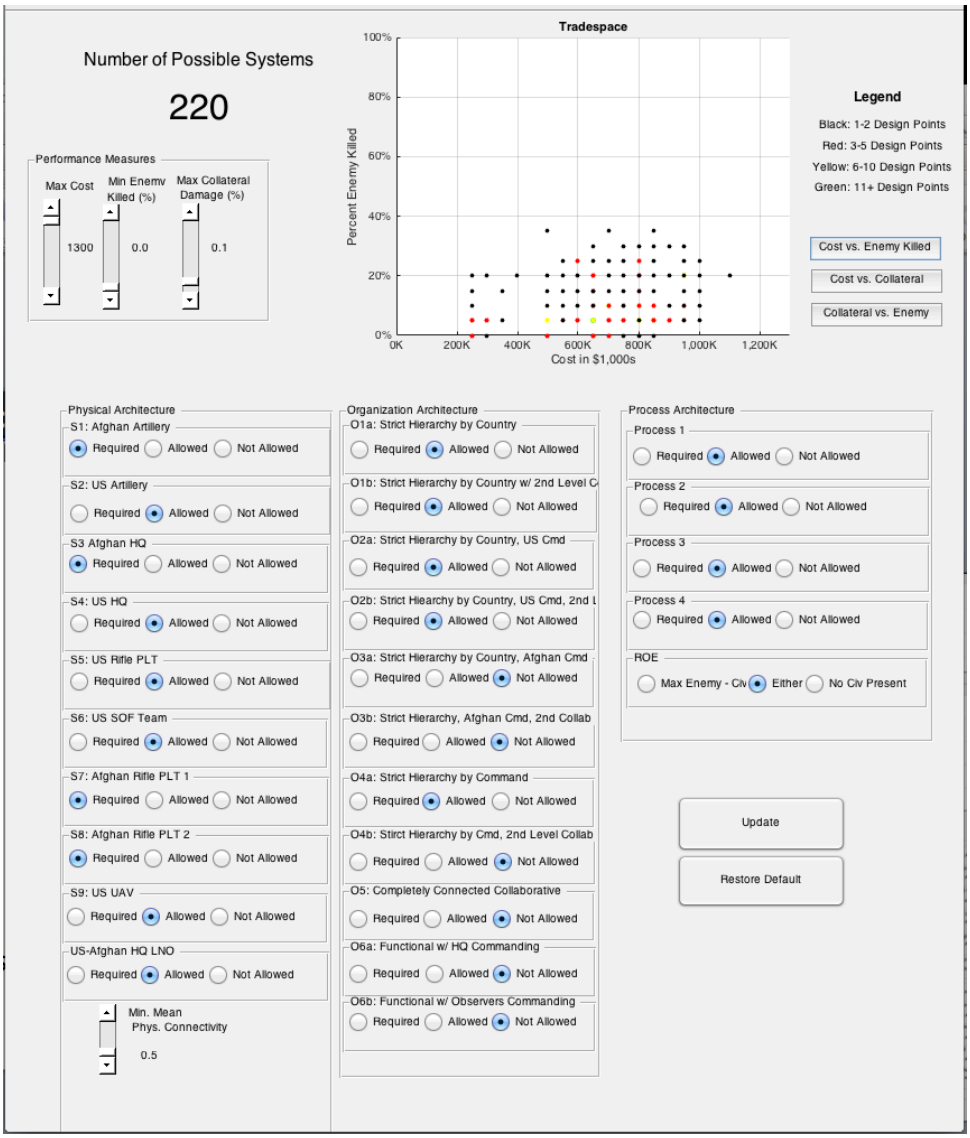


Figure 93. Afghan Forces and Hierarchy Required, 10% PCD

Finally, if the decision-maker still requires his political considerations (including Afghan forces and mandating U.S. Army control), but wishes to improve the PTD at the cost of relaxing PCD, one can see variations in the tradespace. By relaxing the PCD to 11% from 10%, one achieves a potential 45% PTD (up 10% from 35%), although at a higher cost of $850,000 (up from $500,000) as seen in Figure 94. To achieve the 55% PTD achieved without the political considerations at 10% PCD, but with political considerations included, one must raise the maximum PCD to 16%. At this point, there is a point that achieves a PCD of 55% for $700,000 as seen in Figure 95.

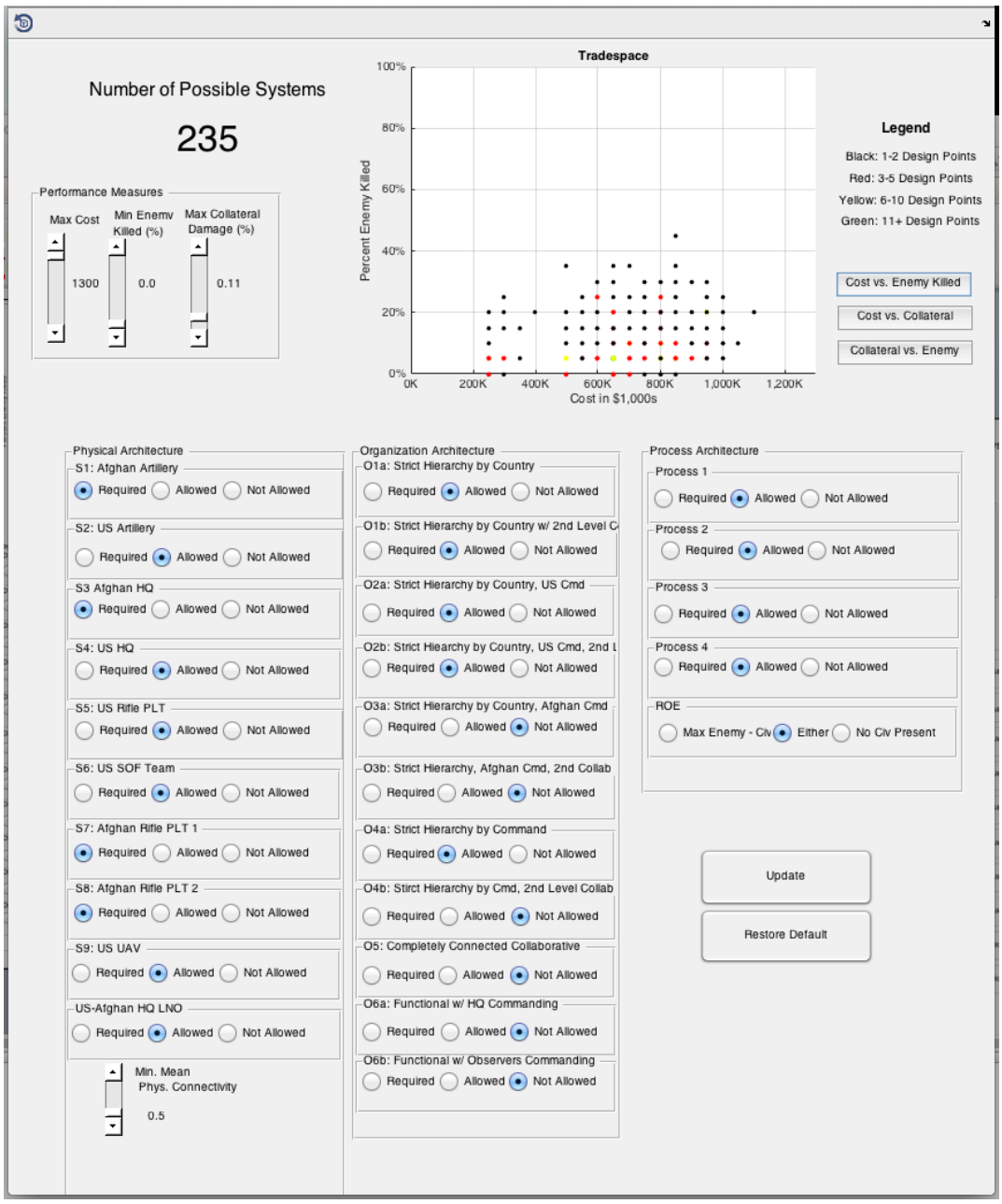


Figure 94. Tradespace 11% PCD with Potential Political Considerations

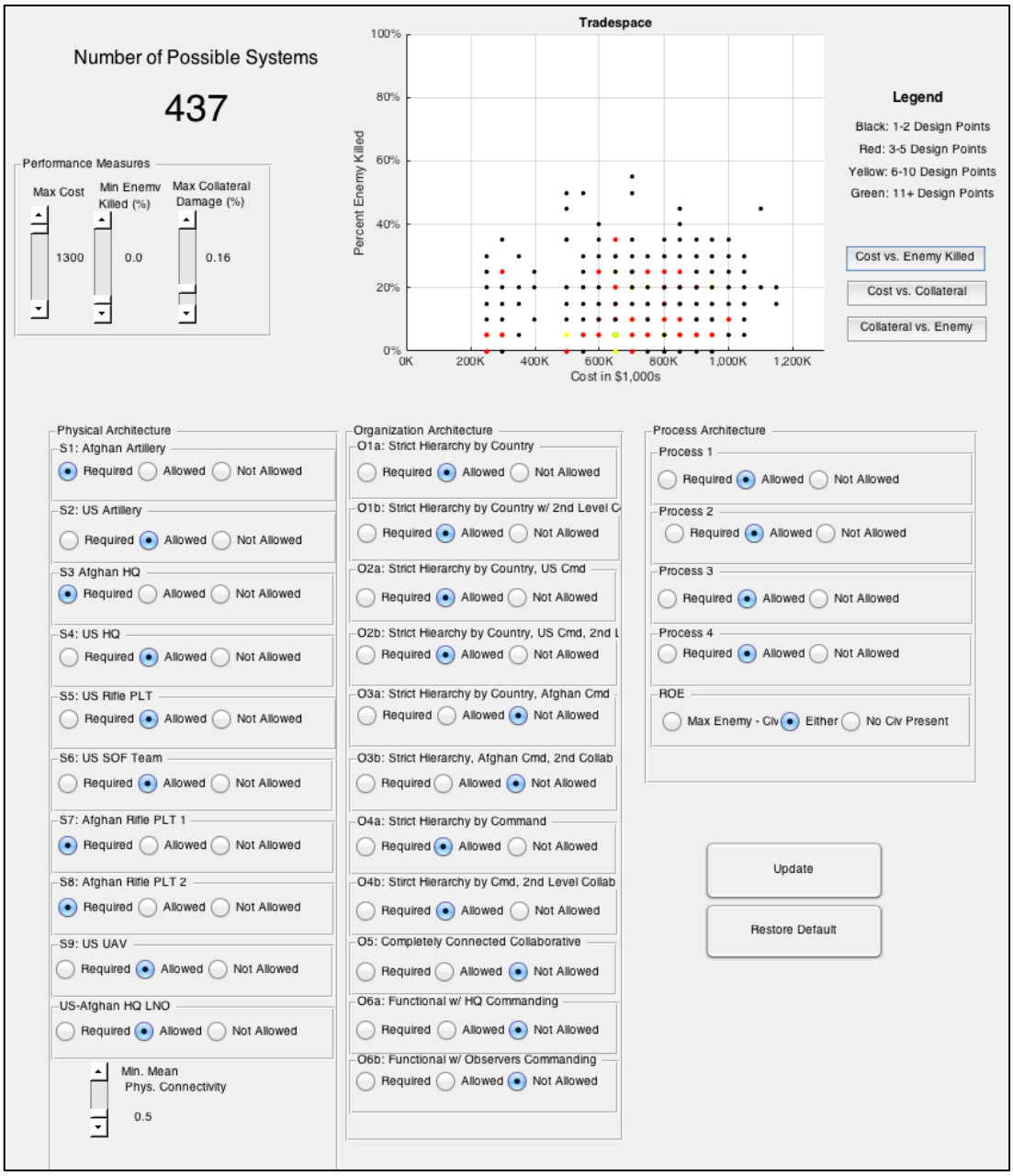


Figure 95. 16% PCD with Potential Political Considerations

The point of this section is, ultimately, not to decide upon a specific IDF SoS design, rather, it is to demonstrate how a tradespace tool may be used in the development of an SoS design, or design criteria. It can help a decision-maker understand his true values, the tradeoffs necessary for the design problem, and potentially, allow operational considerations to be the driving force in design decisions (as opposed to SoS composition).

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Agarwal, Siddhartha, Louis E. Pape, Cihan H. Dagli, Nil K. Ergin, David Enke, Abhijit Gosavi, Ruwen Qin, Dincer Konur, Renzhong Wang, and Ram Deepak Gottapu. 2015. "Flexible and Intelligent Learning Architectures for SoS (FILA-SoS): Architectural Evolution in Systems-of-Systems." *Procedia Computer Science.* 44: 76–85. doi:10.1016/j.procs.2015.03.005.

Ahuja, Ravindra K., Thomas L. Magnanti, and James B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Englewood Cliffs, NJ: Prentice Hall.

Anderson, Dennis J., Tamara J. Brown, and Charles M. Carter. 2013. "System of Systems Operational Availability Modeling." Last modified 24 October. http://www.acq.osd.mil/se/webinars/2013_10_24-SOSECIE-Anderson-Brief.pdf.

Archer, Joshua. 2014. "Identifying Governance Best Practices in Systems-of-Systems Acquisition." CSIS-AM-14-006. Washington, DC: Center for Strategic and International Studies. http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA601872.

Baldwin, W. Clifton, Brian Sauser, and Robert Cloutier. 2015. "Simulation Approaches for System of Systems: Events-based versus Agent Based Modeling." *Procedia Computer Science*. 44: 363–372. doi:10.1016/j.procs.2015.03.032.

Beery, Paul T. 2016. "A Model Based Systems Engineering Methodology for Employing Architecture in System Analysis: Developing Simulation Models Using Systems Modeling Language Products to Link Architecture and Analysis." PhD Dissertation, Naval Postgraduate School.

Bennett, Clayton. Christopher Farris, Paul Foxx, Hughlyn Henderson, Stacy Himes, Corey Kennington, Matthew Mussman, Michael Newman, Maysam Sarfaraz, and Brandon Harwood. 2014. "Operational Energy / Operational Effectiveness Investigation for Scalable Marine Expeditionary Brigade Forces in Contingency Response Scenarios." Master's Thesis, Naval Postgraduate School.

Biltgen, Patrick T., Tommer Ender, and Dimitri N. Mavris. 2006. "Development of a Collaborative Capability-Based Tradeoff Environment for Complex System Architectures." In *44th AIAA Aerospace Sciences Meeting and Exhibit*, 9–12. doi:10.2514/6.2006-728.

Blanchard, Benjamin S., and Wolter J. Fabrycky. 2011. *Systems Engineering and Analysis, Fifth Edition.* New York, NY: Prentice Hall.

Boardman, John, and Brian Sauser. 2006. " System of Systems—The Meaning of *of*." In the *Proceedings of the 2006 IEEE/SMC International Conference on System of Systems Engineering.* Los Angeles, CA: IEEE. doi:10.1109/SYSOSE.2006.1652284.

Bonagrazia-Healey, Viviane; Alain DeLeon, Hand Nguyen, Raymond Chun, David Faulk, Aaron Oostdyk, Victoria Woods, Zachary Crane, Julie Ligman, and Brandon Will. 2014. "Supply Chain Management Model for Modular or Flexible Optimally Manned Ships." Master's Thesis, Naval Postgraduate School.

Box, George E. P., and Norman R. Draper. 1987, *Empirical Model Building and Response Surfaces*. New York, NY: John Wiley & Sons.

Brantley, Mark W., Willie J. McFadden, and Mark J. Davis. 2002. "Expanding the Trade Space: An Analysis of Requirements Tradeoffs Affecting System Design." *Acquisition Review Quarterly*, Winter. http://handle.dtic.mil/100.2/ADA488461.

Buede, Dennis M. 2000. *The Engineering Design of Systems: Models and Methods.* New York, NY: John Wiley & Sons.

Burton, Richard M., Gerardine DeSanctis, and Borge Obel. 2006. *Organizational Design: A Step-by-Step Approach.* New York, NY: Cambridge University Press.

Carlsen, Daniel E. 2008. "Assessment of User-Guided Visual Steering Commands During Trade Space Exploration." Master's Thesis, Pennsylvania State University.

Chattopadhyay, Debarati. 2009. "A Method for Tradespace Exploration of Systems of Systems." Master's Thesis, Massachusetts Institute of Technology.

Chattopadhyay, Debarati, Adam M. Ross, and Donna H. Rhodes. 2008. "A Framework for Tradespace Exploration of Systems of Systems." In *6th Conference on Systems Engineering Research,* Los Angeles, CA.

Chattopadhyay, Debarati, Adam M. Ross, and Donna H. Rhodes. 2009. "Combining Attributes for Systems of Systems in Multi-Attribute Tradespace Exploration." In *7th Conference on Systems Engineering Research*.

Cole, Reggie. 2008. "SoS Architecture." In *Systems of Systems Engineering: Principles and Applications,* edited by Mo Jamshidi, 37–69. Boca Raton, FL: CRC Press.

Cross, Nigel. 2011. *Design Thinking: Understanding How Designers Think and Work.* New York, NY: Bloomsbury.

Daft, Richard L. 1998. *Organization Theory and Design.* Cincinnati, OH: South-Western College.

Dagli, Cihan H., and Nil Kilicay-Ergin. 2009. “System of Systems Architecting.” *System of Systems Engineering: Innovations for the 21st Century,* edited by Mo Jamshidi, 77–100. Hoboken, NJ: Wiley.

Dahmann, Judith, George Rebovich, Ralph Lowry, J. Lane, and Kristen Baldwin. 2011. “An Implementers’ View of Systems Engineering for Systems of Systems.” In *IEEE International Systems Conference 2011*, 212–217. doi:10.1109/SYSCON.2011.5929039.

Dam, Steven H. 2006. *DOD Architecture Framework: A Guide to Applying System Engineering to Develop Integrated, Executable Architectures.* Marshall, VA: System and Proposal Engineering Company.

Davendralingam, Navindran and Daniel DeLaurentis. 2015. “A Robust Portfolio Optimization Approach to System of System Architectures.” *Systems Engineering* 18(3), 269–283. doi:10.1002/sys.21302.

DeLaurentis, Daniel A., En-Pei Han, Tatsuya Kotegawa, and Aaron Sengstacken. 2008. “Utilization of Network Theory for the Enhancement of ATO Air Route Forecast.” In *The 26th Congress of International Council of the Aeronautical Sciences*. Anchorage, AK, 14–19 September. doi:10.2514/6.2008-8944.

Department of Defense (DOD). 2008. “Systems Engineering Guide for Systems of Systems.” Washington, DC. http://www.acq.osd.mil/se/docs/SE-Guide-for-SoS.pdf.

Department of Defense Chief Information Officer (DOD CIO). 2010. “Department of Defense Architecture Framework Version 2.02” Washington, DC. http://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF_v2-02_web.pdf.

Doerry, Norbert, Mark Earnesty, Carol Weaver, Jeff Banko, Jim Myers, Danny Browne, Melissa Hopkins, and Santiago Balestrini. 2014. "Using Set-Based Design in Concept Exploration." *SNAME Chesapeake Section Technical Meeting,* Army-Navy Country Club, Arlington VA. Accessed online May 2016 at http://www.doerry.org/norbert/papers/20140722SBD-CE-final.pdf.

Estefan, Jeff A. 2007. “Survey of Model-Based Systems Engineering (MBSE) Methodologies.” International Council on Systems Engineering Model Based Systems Engineering Focus Group 25. Accessed June 2015 at http://www.visual-process.com/docs/MBSE_Methodology_Survey_RevB.pdf.

Frank, David, Kevin Hogan, Shane Schonhoff, Nicole Becker, Timothy Byram, Richard Kim, Glenna Miller, Scott Myers, and Heather Whitehouse. 2014. “Application of Model-Based Systems Engineering (MBSE) to Compare Legacy and Future Forces in Mine Warfare (MIW) Missions.” Master’s Thesis, Naval Postgraduate School.

Friedenthal, Sanford, Regina Griego, and Mark Sampson. 2007. "INCOSE Model Based Systems Engineering (MBSE) Initiative." Presented at the INCOSE 2007 Symposium. Retrieved 2 February 2015 at: https://www.incose.org/enchantment/docs/07Docs/07Jul_4MBSEroadmap.pdf.

Galbraith Jay, R. 1977 *Organization Design*. Reading, MA: Addison-Wesley.

Garrett, Robert K., Steve Anderson, Neil T. Baron, and James D. Moreland. 2011. "Managing the Interstitials, A System of Systems Framework Suited for the Ballistic Missile Defense System." *Systems Engineering* 14(1), 87–109. doi:10.1002/sys.20173.

Giachetti, Ronald E. 2010. *Design of Enterprise Systems: Theory, Architecture, and Methods*. New York, NY: CRC Press.

Giachetti, Ronald. 2014. "System of Systems." Unpublished PowerPoint slide from lecture, SE 4950: System of Systems Engineering, Naval Postgraduate School, Monterey, CA, July 10.

Giachetti, Ronald E. 2015. "System of Systems Capability Needs Analysis via a Stochastic Network Model," *Naval Engineers Journal* 127(4), 67–79.

Giachetti, Ronald E., Veronica Marcelli, José Cifuentes, and José A. Rojas. 2013. "An Agent‐Based Simulation Model of Human‐Robot Team Performance in Military Environments." *Systems Engineering* 16(1), 15–28. doi:10.1002/sys.21216.

Giachetti, Ronald E. and Clifford Whitcomb. 2016. "Rethinking the Systems Engineering Process in Light of Design Thinking." In *Proceedings of the Thirteenth Annual Acquisition Research Symposium: Volume I:* 48-56. Monterey, CA, May 4-5. Accessed online May 2016 at: https://www.researchsymposium.com/conf/app/researchsymposium/unsecured/file/129/SYM-AM-16-019_Wednesday,%20Vol%201_5-17-2016.pdf.

Gibson, John E., William T. Scherer, and William F. Gibson. 2007. *How to Do Systems Analysis.* Hoboken, NJ: Wiley-Interscience.

Harrison, John A. and Melanie J. Forster. 2003. "Human Systems Integration Requirements In Systems Acquisition." *Handbook Of Human Systems Integration,* edited by Harold R. Booher, 167–200. New York, NY: Wiley.

Holland, Jeffery P. 2013. "Engineered Resilient Systems (ERS) Overview." Accessed June. 2015 at: http://www.defenseinnovationmarketplace.mil/resources/ERS_Overview_2DEC2013-Final.pdf.

IDEO. 2016. “About IDEO.” Accessed April 5. https://www.ideo.com/about/.

International Council on Systems Engineering (INCOSE). 2010. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. INCOSE-TP-2003-002-03.2. San Diego, CA: International Council on Systems Engineering.

International Council on Systems Engineering (INCOSE). 2015. “SE Transformation.” Accessed August 15. http://www.incose.org/about/strategicobjectives/transformation.

Jamshidi, Mo. 2008. *Systems of Systems Engineering: Principles and Applications*. Boca Raton, FL: CRC Press.

Jamshidi, Mohammad. 2009. *System of Systems Engineering: Innovations for the Twenty-First Century*. Hoboken, NJ: Wiley.

Jepperson, Dustin B. 2013. “Using Model Based Systems Engineering and the Systems Modeling Language to Develop Space Mission Area Architectures.” Master’s Thesis, Naval Postgraduate School.

Joint Chiefs of Staff (JCS). 2011. *Joint Operations.* (JP 3–0). Washington, DC: Department of Defense.

Kaymal, Turgut. 2013. “Assessing the Operational Effectiveness of a Small Surface Combat Ship in an Anti-Surface Warfare Environment.” Master’s Thesis, Naval Postgraduate School.

Keating, Charles B. 2009. “Emergence in System of Systems.” *System of Systems Engineering: Innovations for the 21st Century*, edited by Mo Jamshidi, 169–190. Hoboken, NJ: Wiley.

Keeney, Ralph L. 1992. *Value Focused Thinking: A Path to Creative Decision Making.* Cambridge, MA: Harvard University Press.

Kenley, C. Robert. 2015. “Synthesizing and Specifying Architectures for Systems of Systems.” Presentation to the System of Systems Engineering Collaborators Information Exchange, April 28. Accessed 10 April 2016. http://www.acq.osd.mil/se/webinars/2015_04_28_Kenley-SOSECIE-brief.pdf.

Kenley, C. Robert, Timothy M. Dannenhoffer, Paul C. Wood, and Daniel A. DeLaurentis. 2014. “Synthesizing and Specifying Architectures for System of Systems.” Paper Presented at the 24th Annual INCOSE International Symposium (IS2014), Las Vegas, NV, 30 June – 3 July 2014. Available at: http://web.ics.purdue.edu/~ckenley/pubs/2014.pdf.

Kernstine, Kemp H. 2012. “Design Space Exploration of Stochastic System-of-Systems Simulations Using Adaptive Sequential Experiments.” PhD Dissertation, Georgia Institute of Technology.

Kernstine, Kemp H. 2013. “Inadequacies of Traditional Exploration Methods in Systems-of-Systems Simulations.” *IEEE Systems Journal* 7(4), 528–536. doi: 10.1109/JSYST.2013.2252864.

Kewley, Robert H. and Marc Wood. 2012 “Federated Simulation for System of Systems Engineering.” *Engineering Principles of Combat Modeling and Distributed Simulation*, edited by Andreas Tolk, 765–810. Hoboken, NJ: Wiley.

Kleijnen, Jack PC, Susan M. Sanchez, Thomas W. Lucas, and Thomas M. Cioppa. 2005. “State-of-the-Art Review: A User’s Guide to the Brave New World of Designing Simulation Experiments.” *INFORMS Journal on Computing* 17(3), 263–289. no. 3 (2005): 263–289. doi:10.1287/ijoc.1050.0136.

Lane, Jo Ann, and Tim Bohn. 2013. “Using SysML Modeling to Understand and Evolve Systems of Systems.” *Systems Engineering* 16(1), 87–98. doi:10.1002/sys.21221.

Law, Averill M. 2008. *Simulation Modeling and Analysis*. New Delhi, India: Tata McGraw Hill.

Lichtenstein, Sarah and Slovic, Paul. 2006. *The Construction of Preference*. New York, NY: Cambridge University Press.

Macal, Charles M., and Michael J. North. 2005. “Tutorial on Agent-Based Modeling and Simulation.” In Proceedings of the 37th Winter Simulation Conference, 2–15.

MacCalman, Alex, Hyangshim Kwak, Mary McDonald, Steve Upton, Coleman Grider, Robert Hill, Hunter Wood, and Paul Evangelista. 2015. “Illuminating Tradespace Decisions Using Efficient Experimental Space-Filling Designs for the Engineering Resilient System Architecture.” Technical Report DSE-R-1501. West Point, NY: United States Military Academy Operations Research Center.

Maier, Mark W. 1998. “Architecting Principles for Systems-of-Systems.” *Systems Engineering* 1(4), 267–284. doi:10.1002/(SICI)1520-6858.

Maier, Mark W. 2015. “The Role of Modeling and Simulation in System of Systems Development.” *Modeling and Simulation Support for System of Systems Engineering Applications*, edited by Larry B. Rainey and Andreas Tolk, 11–41. Hoboken, NJ: Wiley.

Maier, Mark W. and Rechtin, Eberhardt. 2009. *The Art of Systems Architecting*. Boca Raton, FL: CRC Press.

March, James G., and Herbert Alexander Simon. 1958. *Organizations.* New York, NY: Wiley.

Mokhtarpour, Behrokh, and Jerrell Stracener. 2014. "A Conceptual Methodology for Selecting the Preferred System of Systems." *IEEE Systems Journal* 99, 1–7. doi:10.1109/JSYST.2014.2352332.

Montgomery, Douglas, C. 2005. *Design and Analysis of Experiments: 6th Edition.* Hoboken, NJ: John Wiley and Sons, Inc.

Mour, Ankur, C. Robert Kenley, Navindran Davendralingam, and Daniel DeLaurentis. 2013. "Agent-Based Modeling for Systems of Systems." Presented at the 23nd Annual International Symposium of the International Council of Systems Engineering, at Philadelphia, PA.

Nelson, Harold G., and Erik Stolterman. 2003. *The Design Way.* Englewood Cliffs, NJ: Educational Technology Publication.

Newman, M.E.J. 2010. *Networks: An Introduction.* Oxford, United Kingdom: Oxford University Press.

Norman, Douglas O., and Michael L. Kuras. 2006. "Engineering Complex Systems." *Complex Engineered Systems*, edited by Dan Braha, Ali A. Minai, and Yaneer Bar-Yam, 206–245. New York, NY: Springer.

North Atlantic Treaty Organization (NATO). 2015. "NATO Glossary of Terms and Definitions (English and French) (AAP-6(2015))." Brussels, Belgium: NATO. Accessed April 14 2016 at https://nso.nato.int/nso/nsdd/ListPromulg.html.

Object Management Group (OMG). 2015. "Systems Modeling Language." Accessed August 15. http://www.omgsysml.org/.

Object Management Group (OMG). 2016. "Unified Profile For the Department of Defense Architecture Framework (DoDAF) and the Ministry of Defence Architecture Framework (MODAF)." Accessed April 10. http://www.omg.org/spec/UPDM/.

Owen, Guillermo. 2013. *Game Theory.* Bingley, UK: Emerald Publishing.

Pan, Xing, Baoshi Yin, and Jianmi Hu. 2011. "Modeling and Simulation for SoS Based on the DoDAF Framework." 9th International Conference on Reliability, Maintainability and Safety (ICRMS), 1283–1287. doi:10.1109/ICRMS.2011.5979468.

Parnell, Gregory S., Patrick J. Driscoll, and Dale L. Henderson. 2011. *Decision Making in Systems Engineering and Management, Second Edition*. Hoboken, NJ: Wiley.

Paulo, Eugene; Beery, Paul; MacCalman, Alex. "Illuminating the Tradespace." *Systems Engineering,* Forthcoming.

Pennsylvania State University Applied Research Laboratory (PSU-ARL). 2015. "Trade Space Exploration." Accessed 27 January. http://www.atsv.psu.edu/index.html.

Pernin, Christopher G., Elliot Axelband, Jeffrey A. Drezner, Brian B. Dile, John Gordon IV, Bruce J. Held, K. Scott McMahon, Walter L. Perry, Christopher Rizzi, Akhil R. Shah, Peter A. Wilson, and Jerry M. Sollinger. 2012. *Lessons from the Army's Future Combat Systems Program*. Santa Monica, CA: RAND Arroyo Center.

Pisani, Christopher R. 2013. "Linking Combat Systems Capabilities and Ship Design Through Modeling and Computer Simulation." Master's Thesis, Naval Postgraduate School.

Rainey, Larry B. and Tolk, Andreas. 2015. *Modeling and Simulation Support for System of Systems Engineering Applications.* Hoboken, NJ: Wiley.

Rao, Madwaraj; Ramakrishnan, Sreeram; and Dagli, Cihan. 2008. "Modeling and Simulation of Net Centric Systems of Systems Using Systems Modeling Language and Colored Petri-Nets: A Demonstration Using the Global Earth Observation System of Systems." *Systems Engineering* 11(3), 203–220. doi:10.1002/sys.20095.

Reingold, Omer. 2008. "Undirected Connectivity in Log-Space." *Journal of the ACM (JACM)* 55(4), 17:1-17:24. doi: 0.1145/1391289.1391291.

Reynolds, Craig W. 1987. "Flocks, Herds and Schools: A Distributed Behavioral Model." *ACM Siggraph Computer Graphics* 21(4), 25–34. doi:10.1145/37401.37406.

Ross, Adam M. 2006. "Managing Unarticulated Value: Changeability in Multi-Attribute Tradespace Exploration." PhD dissertation, Massachusetts Institute of Technology.

Ross, Adam M., and Daniel E. Hastings. 2005. "The Tradespace Exploration Paradigm." *INCOSE International Symposium*, 15, 1706–1718. doi:10.1002/j.2334-5837.2005.tb00783.x.

Ross, Adam M. and Donna H. Rhodes. 2015. "An Approach for System of Systems Tradespace Exploration." *Modeling and Simulation Support for System of Systems Engineering Applications* edited by Larry P. Rainey and Andreas Tolk,75-98. Hoboken, NJ: Wiley.

Rowden, Thomas, Peter Gumataotao, and Peter Fanta. 2015. "Distributed Lethality." *Proceedings Magazine*. 141/1/1,343. http://www.usni.org/magazines/proceedings/2015-01/distributed-lethality.

Sage, Andrew P., and Steven M. Biemer. 2007. “Processes For System Family Architecting, Design, and Integration.” *IEEE Systems Journal* 1(1), 5–16. doi:10.1109/JSYST.2007.900240.

Sanchez, Susan M., and Hong Wan. 2012 "Work Smarter, Not Harder: A Tutorial on Designing and Conducting Simulation Experiments." *Proceedings of the 2012 Winter Simulation Conference (WSC).* 1-15. doi:10.1109/WSC.2012.6465307.

Singer, David J., Norbert Doerry, and Michael E. Buckley. 2009. “What is Set Based Design?” *Naval Engineers Journal.* 121(4): 31-43. doi: 10.1111/j.1559-3584.2009.00226.x.

Sitterle, Valerie B., Dane F. Freeman, Simon R. Goerger, and Tommer R. Ender. 2015. “Systems Engineering Resiliency: Guiding Tradespace Exploration within an Engineered Resilient Systems Context.” *Procedia Computer Science* 44, 649–658. (2015): 649–658. doi:10.1016/j.procs.2015.03.013.

Sobek II, Durward K., Allen C. Ward, and Jeffrey K. Liker. 1999. "Toyota's Principles of Set-Based Concurrent Engineering." *MIT Sloan Management Review* 40, no. 2: 67.

Sokolowski, John A. and Catherine M. Banks. 2011. *Principles of Modeling and Simulation: A Multidisciplinary Approach*. Hoboken, NJ: John Wiley & Sons.

Srivastava, Tina P., Victor L. Piper, and Jose M. Arias. 2014. “Future Combat Systems Case Study for Analysis of System of Systems Approach.” INCOSE International Symposium, 22: 1947 – 1966. doi:10.1002/j.2334-5837.2012.tb01448.x.

Statnikov, Roman B. and Joseph B. Matusov. 2002. *Multicriteria Analysis in Engineering.* Boston, MA: Kluwer Academic Publishers.

Stump, Gary, Mike Yukish, Jay D. Martin, and T. W. Simpson. 2004. “The ARL Trade Space Visualizer: An Engineering Decision-Making Tool.” *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference* 30. doi:10.2514/6.2004-4568.

Thompson, Bruce, Craig Lawton, and Kimberly Welch. 2011. “Using the System-of-Systems Analysis Toolset (SOSAT) for Systems Engineering Analysis.” Accessed April 2015 at http://www.ndia.org/Divisions/Divisions/SystemsEngineering/Documents/Committees/System%20of%20Systems%20Committee/new%202011/6.February_11.SoS.Analysis.Tool.pdf.

U.S. Army. 2006. *The Infantry Rifle Company.* (FM 3–21.10). Washington, DC: Department of the Army.

U.S. Army. 2014. *The U.S. Army Operating Concept: Win in a Complex World* (TRADOC Pamphlet 525–3-1). Washington, DC: Department of the Army.

Vieira Jr., Hélcio, Susan M. Sanchez, Karl Heinz Kienitz, and Mischel Carmen Neyra Belderrain. 2011. "Improved Efficient, Nearly Orthogonal, Nearly Balanced Mixed Designs." *Proceedings of the 2011 Winter Simulation Conference (WSC). 3600-3611.* doi: 10.1109/WSC.2011.6148054.

Wang, Renzhong. 2007. "Executable System Architecting Using Systems Modeling Language in Conjunction with Colored Petri Nets – A Demonstration Using the GEOSS Network Centric System." Master's Thesis, Missouri University of Science and Technology.

Wang, Wenguang, Andreas Tolk, and Weiping Wang. 2009. "The Levels of Conceptual Interoperability Model: Applying Systems Engineering Principles to M&S." *Proceedings of the 2009 Spring Simulation Multiconference*. http://dl.acm.org/citation.cfm?id=1655398.

Welch, Savannah G. 2011. "Investigating The Link Between Combat System Capability And Ship Design." Master's Thesis, Naval Postgraduate School.

White, K. Preston. 1998. "Systems Design Engineering." *Systems Engineering* 1(4), 285–302. doi:10.1002/(SICI)1520-6858.

Wymore, A. Wayne. 1993. *Model-Based Systems Engineering*. New York, NY: CRC Press.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California